(2)

非同期回路の設計とマイクロプロセッサの設計に関する研究

平成8年6月

井上訓行

目　　次

第１章　まえがき

論理システムの基本構成要素は論理ゲートである．論理ゲートと配線から成る論理回路はその動作から同期回路と非同期回路に分けられる．同期回路は外部から与える同期信号（クロックパルス）に回路の動作を同期させる方式であり，非同期回路は同期信号を用いず回路の遅延に依存して動作を進める方式である．

同期回路はクロックパルスに同期させることによって素子や配線のばらつきによるタイミングの問題を回避できるので回路構成が簡単になり，マイクロプロセッサなどの大規模なシステムではほとんど同期回路が用いられる．その基本構成要素はゲートとフリップフロップであり，フリップフロップにクロックパルスを与えることによって同期がとられる．

一方，非同期回路は高速回路が実現できるが，ゲートと配線の遅延からハザードやレースの問題が生じ[1]，回路構成が複雑になるため，システム全体を非同期回路で構成することは難しい．しかし，独立に動作するシステムの相互接続や同期システムの基本構成要素は非同期回路である．前者にはアービタ[2]などがあり，後者にはフリップフロップがある．フリップフロップはそれ自体はクロックを特別の入力とはみなさない非同期回路である．すなわち非同期回路

で構成されたフリップフロップが，クロックパルスを特別の同期信号とみなすことによって同期回路の構成要素となり，複雑な論理システムを構成している．

フリップフロップは広く使用されており，多くの回路が知られている[3]～[6]．しかし，同期回路の構成要素として，その動作は論じられているが[4][5]，非同期回路理論にもとづいて理論的，系統的に設計されることは比較的少ない．このため，論理的に同じ回路が別名で呼ばれたり（Ｄフリップフロップのマスタスレーブ型とエッジトリガ型など[7]），論理的に異なる動作をするものが同名で呼ばれたり（ある種のＪＫフリップフロップなど[3]）している．これらの異同を明確にしておく必要がある．

半導体集積回路技術の進展により，複雑な論理システムが比較的容易に開発できるようになっている．特に高密度のプログラム可能なＬＳＩが利用可能になり，大学等の研究室内においても必要な機能を持つＩＣを開発できるようになってきている[8][9]．これらはほとんど同期回路である．ＣＡＤシステムを用いた大規模な同期回路を設計するのための開発システムを構築する必要が生じている．また，情報工学教育においてもＬＳＩ設計教育が重要になり[9]～[12]，このための実験実習環境を整備する必要がある．

本文では，まず，ＳＲラッチによる非同期回路の設計法について

述べ，次にその設計法にもとづいて同期回路の構成要素となるフリップフロップを理論的に設計する．最後にフリップフロップを構成要素とした同期回路で複雑なシステムを開発するためのシステムについて述べる．

本論文の構成

第１章では本研究の背景と目的を明らかにし，本研究の内容を概説した．

第２章ではＳＲラッチによる非同期回路の設計法について述べる．非同期回路は組合せ回路にフィードバックループと遅延素子を挿入することによって実現されるが[1]，この遅延素子の代わりに，ループにＮＡＮＤまたはＮＯＲゲート２個をたすきがけにしたＳＲラッチを挿入して設計することもできる[13][14]．この方法はＳＲラッチによる設計法と呼ばれ，その特徴は非同期回路の設計における最大の問題の１つであるハザードを，余分のゲートを追加することなく除去できることである[14]．

第３章では非同期回路理論にもとづいて同期回路の構成要素となるフリップフロップを理論的，系統的に設計する．その結果広く使用されているほとんどのＤフリップフロップとＪＫフリップフロップが設計され，各々の論理的特性が明らかにされる[3]．さらに特殊

なフリップフロップとしてエッジセンシングマスタスレーブ型フリップフロップ[3][4]とダブルエッジトリガ型フリップフロップ[3][15]が設計され，第２章で述べたＳＲラッチによる設計法が有効であることが示される．

第４章ではフリップフロップを基本構成要素とする同期回路の設計開発システムについて述べる．最も複雑な論理システムであるマイクロプロセッサを設計対象としＦＰＧＡ（Field Programmable Gate Array)上にマイクロプロセッサを実現するためのシステムが構築されている[9][10]．このシステムはすべてＲＡＭ構造のＬＳＩ（ＦＰＧＡとメモリ）から成る特徴のあるシステムであり，計算機工学の実験実習教育に応用され，多数のマイクロプロセッサが開発されている．

最後に第５章では本論文で得られた結果を要約する．

参　考　文　献

[1] S.H.Unger: Asynchronous Sequential Switching Circuits, Wiley-Inter-Science, New York(1969).

[2] J.Calvo, J.I.Acha and M.Valencia: Asynchronous Modular Arbiter, IEEE Transactions on Computers, Vol.C-35, No.1, pp. 67-70(1986).

[3] 井上訓行，奥川峻史：非同期回路理論によるラッチとフリップフロップの設計，電子通信学会技術研究報告,CPSY86-58(1987).

[4] F.J.Hill and G.R.Peterson: Introduction to Switching Theory and Logical Design, 3rd ed., John Wiley & Sons, New York(1981).

[5] S.Muroga(室賀三郎,笹尾勤訳)：論理設計とスイッチング理論，共立出版(1981).

[6] Texas Instruments: The Bipolar Digital Integrated Circuits Data Book(1976).

[7] S.Okugawa and N.Inoue: Systematic Design of D Flip-flops Using Two State Variables, Electronics Letters Vol.23, No. 17, pp.909-910(1987).

[8] 冨田昌宏，澄川文徳，菅沼直昭，平野浩太郎：汎用エンジン ＲＭ－Ⅱの構成，情報処理学会論文誌,Vol.35,No.4,pp.636-644 (1994).

[9] 井上訓行：ＦＰＧＡを用いたマイクロプロセッサ開発システムとその教育への応用，電子情報通信学会技術研究報告,CPSY94-58(1994).

[10] 井上訓行：計算機アーキテクチャ教育とその支援環境，情報処理学会論文誌，Vol.35,No.2,pp.155-163(1994).

[11] 神原弘之，安浦寛人：計算機教育用マイクロコンピュータの開発とその応用，情報処理，Vol.33,No.2,pp.118-127(1992).

[12] 末吉敏則：教育へのＦＰＧＡ応用例，情報処理,Vol.35,No.6,pp.519-529(1994).

[13] D.B.Armstrong, A.D.Friedman and P.R.Menon: Realization of Asynchronous Sequential Circuits Without Inserted Delay Elements, IEEE Transactions on Computers,Vol.C-17, No.2,pp.129-134(1968).

[14] 井上訓行，奥川峻史：ＳＲラッチによる非同期回路の設計,情報処理学会論文誌,Vol.28,No.10,pp1051-1059(1987).

[15] S.H.Unger: Double-Edge-Triggered Flip-flops, IEEE Transactions on Computers,Vol.C-30,No.6,pp.447-451(1981).

第２章　ＳＲラッチによる非同期回路の設計

２．１　はじめに

順序回路は遷移表が与えられたとき，各状態に状態変数を割り当て，状態変数の論理式を導くことによって設計される．非同期式順序回路ではレースがないように状態を割り当て，ハザードがないよう（ハザードフリー）に論理式を求めなければならない[1].

ＳＲラッチは単独で用いられることは少ないが，非同期式順序回路の基本構成要素として広く利用されている．ＳＲラッチによる設計法は順序回路の状態変数１つにＳＲラッチ１個を割り当て，各ラッチのＳ，Ｒ入力を求めて非同期回路を設計する方法である[2][3].この方法では，論理式をハザードフリーに導く必要はない．入力にハザードがある場合のＳＲラッチの動作については，明確にされているとはいえない.

本章では，ＳＲラッチを解析しＳＲ＝０のもとではＳ，Ｒ入力が同時に変化しても安定に動作し，Ｓ，Ｒ入力にハザード（ＮＡＮＤラッチでは静的０ハザード，ＮＯＲラッチでは静的１ハザード）があっても出力に影響しないことを明確にする．この結果Ｓ，Ｒ入力を２段回路（１段はラッチのゲートを利用するので実質１段回路）で構成すれば，ＮＡＮＤラッチでは静的１ハザードが，ＮＯＲラッチでは

静的０ハザードがなく，いずれにも動的ハザードはない．したがってハザードの問題を生じないことがわかる．即ち，ＳＲラッチによる設計法では，ＳＲフリップフロップを使って同期回路を設計するのと，状態割当てを除いて，同じ方法で非同期回路が設計できることになる[3].

つぎに，ＰＬＡやＲＯＭなどのアレイロジックを用いて非同期式順序回路を実現する方法を考察する．ＡＮＤ-ＯＲ２段回路に存在する静的１ハザードはＳＲラッチによって吸収されるので，アレイの中にＳＲラッチを内蔵したＰＬＡを考える．このＰＬＡはＯＲアレイの出力部にたすきがけ回路を入れることによって，外部に付加回路を付けずにＳＲラッチの機能を持つ．したがって，ＳＲラッチによる設計法をそのまま適用して非同期式順序回路を実現できる．最後にＲＯＭによる非同期式順序回路の実現について述べる．

２．２　非同期式順序回路の設計

順序回路は，現在の入力をＸ，現在の状態をＱ，出力をＺ，次の状態を$Q^{(1)}$とすると

$$Z = \omega\ (X, Q)$$

$$Q^{(1)} = \delta\ (X, Q) \qquad (2.1)$$

で定義される．ここでωは出力関数，δは遷移関数である．入力，出力，状態はそれぞれ n，m，q 個の２値変数の組み合わせで表され

$$X = (x_1, x_2, \cdots x_n)$$

$$Z = (z_1, z_2, \cdots z_m) \qquad (2.2)$$

$$Q = (y_1, y_2, \cdots y_q)$$

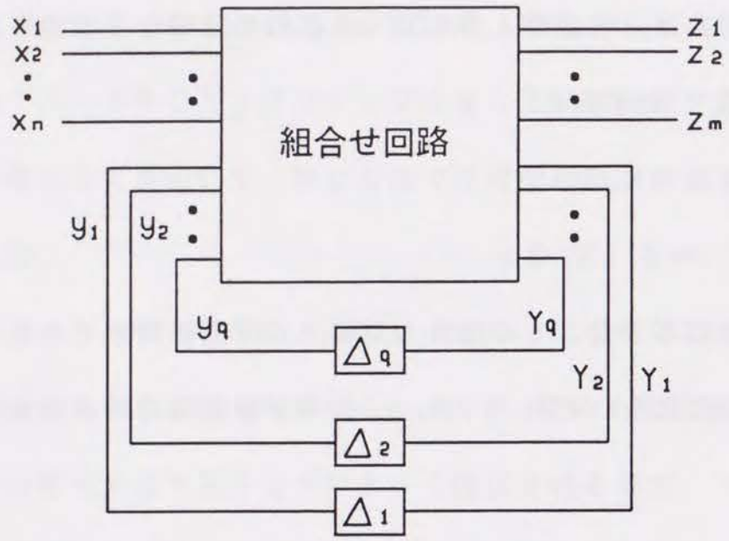
と書くことができる．x_i(i=1～n)，z_j(j=1～m)，y_k(k=1～q)はそれぞれ入力変数，出力変数，状態変数である．(2.1)式をこれらの変数を用いて

$$z_j = g_j(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_q)$$

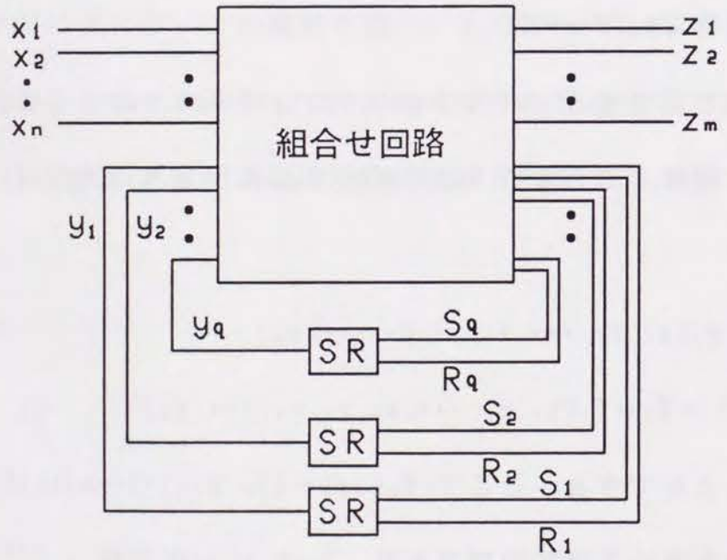
$$y_i^{(1)} = Y_i = f_i(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_q) \qquad (2.3)$$

と表すことができる．ここで f_i(i=1～q)，g_j(j=1～m)は状態遷移および出力を決める論理関数であり，Y_iは y_iの次の値 $y_i^{(1)}$である．

非同期式順序回路では(2.3)式は図２．１(a)の形で実現される．組合せ回路は(2.3)式の f_iと g_jを実現する（n+q)入力（m+q)出力の論



(a) 遅延素子による構成



(b) ＳＲラッチによる構成

図２．１ 非同期式順序回路

理回路である.

ＳＲラッチを用いて Y_iを表すと図２．１(b)の形で実現される.

S_i, R_iは

$$S_i = s_i(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_q)$$

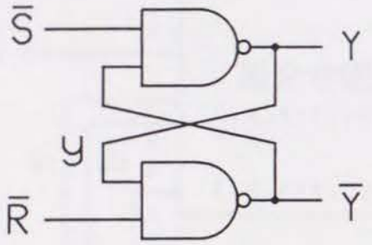
$$R_i = r_i(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_q) \qquad (2.4)$$

で表される論理式である(i=1～q).

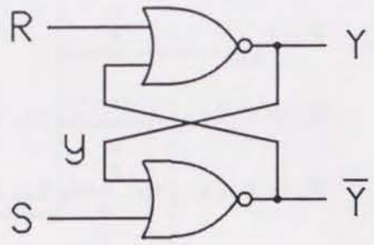
本章ではＳＲラッチを用いて非同期式順序回路を図２．１(b)の形で実現する方法について述べる[3]. 本章で扱う非同期式順序回路は, 基本モードで動作し入力は同時に２つ以上変化しない (Single Input Change)と仮定する[1]. また, クリティカルレースがないように状態割当されており, もし本質的ハザードがあれば物理的に回避できるものとする.

２．３ ＳＲラッチ

非同期回路の基本要素となるＳＲラッチについて述べる. 図２．２(a),(b)に示すようにＳＲラッチはＮＡＮＤゲートまたはＮＯＲゲート２個で構成され, それぞれＮＡＮＤラッチ, ＮＯＲラッチと呼ばれている. このラッチの特性表と励起表を図２．２(c),(d)に示す. 図２．２(d)は同期回路で使用されるＳＲフリップフロップの励起表とまったく同じである.



(a) ＮＡＮＤラッチ



(b) ＮＯＲラッチ

S R	NAND Y $\bar{Y}$	NOR Y $\bar{Y}$
0 0	y $\bar{y}$	y $\bar{y}$
0 1	0 1	0 1
1 0	1 0	1 0
1 1	1 1	0 0

(c) 特性表

y	Y	S	R
0	0	0	d
0	1	1	0
1	0	0	1
1	1	d	0

(d) 励起表

図２．２ ＳＲラッチ

２．３．１ ＳＲラッチの論理的解析

ＳＲラッチを設計に利用するため，まず論理的に解析する．図２．２(a),(b)は図２．３(a),(b)のように書き換えることができ，そのときの遷移表は図２．２(c),(d)となる．ＹはＮＡＮＤラッチでは，

$$Y = S + \overline{R}\, y \qquad (2.5)$$

ＮＯＲラッチでは

$$Y = \overline{R}\, S + \overline{R}\, y \qquad (2.6)$$

となる．

ＳＲ＝１のとき次の問題を生ずる．

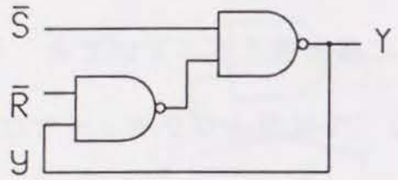
(1) ＳＲが 11→00と変化したとき，次の状態が確定しない（図２．３(c)(d)）．基本モードで考えると，回路の入力は１つしか変化しないが，回路の内部ではＳとＲが同時に変化する可能性はある．

（ＳＲが 00→11，01→10，10→01 と変化しても問題はない．）

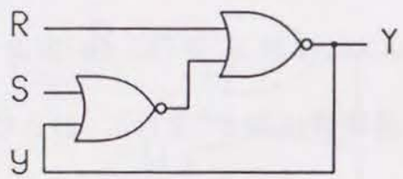
(2) 図２．２(a),(b)においてＳＲ＝１のとき，安定状態ではあるが，出力はＮＡＮＤラッチでは両方１，ＮＯＲラッチでは両方０となり，出力の相補関係は保たれない（図２．２(c)）．

(3) (2.5)式(2.6)式からわかるようにＮＡＮＤラッチとＮＯＲラッチの論理式が異なる．

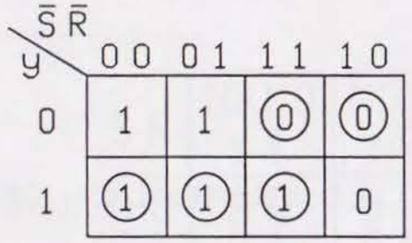
以上の理由からふつうＳＲ＝０を満たすように使用される．このとき(2.5)式(2.6)式は



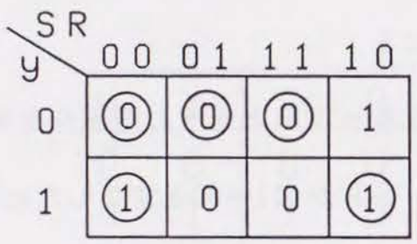
(a) ＮＡＮＤラッチ



(b) ＮＯＲラッチ



(c) ＮＡＮＤラッチの遷移表



(d) ＮＯＲラッチの遷移表

図２．３　ＳＲラッチ

$$Y = S + \overline{R}\,y, \quad SR = 0 \tag{2.7}$$

となる．

しかし，ＳＲラッチそのものはＳＲ＝１でも動作するので，上記(1)(2)(3)を考慮した上であればＳ，Ｒとも１になるときにも使用できる．フリップフロップではこのような使い方がされているものも多い[4]．

２．３．２　遅れを考慮した場合の解析

図２．２(a)のラッチにゲートの遅れを考慮し，ゲートと遅延に分離すると図２．４(a)のように書くことができ，励起表は図２．４(b)になる[5]．ここでは簡単のため遅れは純遅延のみを考える．

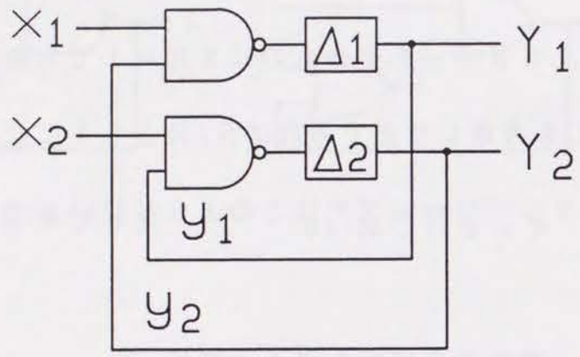
［Ｉ］　入力が同時に変化する場合

① x_1x_2 が 01→10(10→01) のとき $\Delta_1+\Delta_2$ 後に出力は安定状態に落ち着く．

② x_1x_2 が 11→00 のとき $\max(\Delta_1,\Delta_2)$ 後に出力11に落ち着く．

③ x_1x_2 が 00→11 のとき，図２．５に示すように周期 $\Delta_1+\Delta_2$ で発振状態になり，落着き先は決まらない．

この様な状態はメタステーブルと言われ，ディジタルシステムでは避けられない．アービタなどではメタステーブルが重要な問題となるので，慣性遅延のある場合も含めて研究されている[6][7]．



(a) 回路図

y1 y2 \ x1 x2	00	01	11	10
0 0	11	11	11	11
0 1	11	11	(01)	(01)
1 1	(11)	10	00	01
1 0	11	(10)	(10)	11

(b) (a)の遷移表

図２．４ 遅延を考慮したＮＡＮＤラッチ

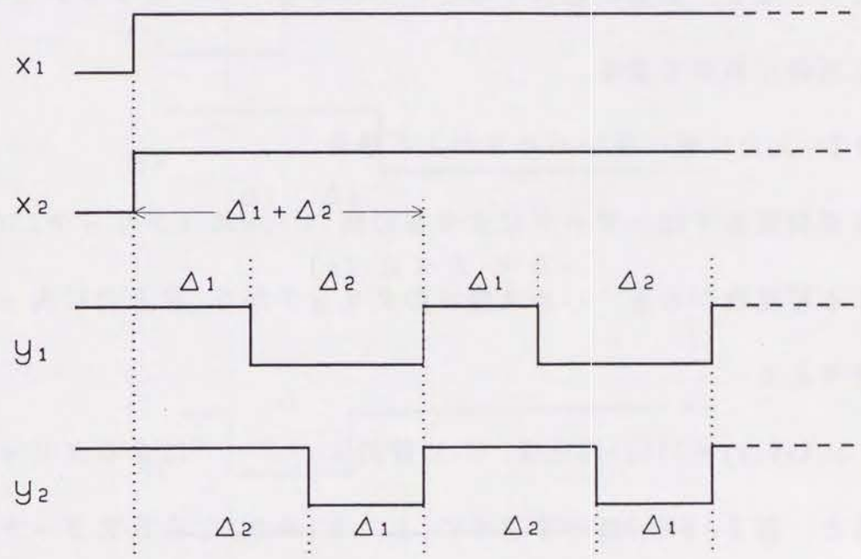


図２．５ タイミング図
（x_1x_2が 00→11 と同時に変化したとき）

安定に動作するためには x_1が変化したあとΔ_1以上，x_2が変化したあとΔ_2以上遅れて他方が変化しなければならない．以上の解析からゲートの遅れを考慮しても $x_1x_2=00$を除いて（図２．２(a)ではＳＲ＝０なら）安定に動作することがわかる．ＮＯＲラッチについても同様に解析できる．

［II］ 入力に幅の狭いパルスが入る場合

非同期回路ではハザードにより幅の狭いパルス（グリッチ）が発生する可能性がある．パルス幅Δのグリッチが S，R 入力に入った場合を考える．

(1) $x_1x_2y_1y_2=1101$ のとき，x_1に静的１ハザードによるグリッチが入ると，図２．６(a)に示すように $\Delta>\Delta_1+\Delta_2$ならＳＲラッチの内容が反転し，安定状態になる．$\Delta<\Delta_1+\Delta_2$なら図２．６(b)に示すように Y_1, Y_2は周期 $\Delta_1+\Delta_2$で発振し，メタステーブルになる．

x_2に静的１ハザードによるグリッチが入っても $y_1=0$であるから Y_2には抜けない．

$x_1x_2y_1y_2=1110$ のときも同様である．

(2) $x_1x_2y_1y_2=1001$ のとき x_1に静的１ハザードによるグリッチが入るとΔ_1後に Y_1に出る．x_2に静的０ハザードによるグリッチが入っても y_1が０であるから出力には出ない．

$x_1x_2y_1y_2=0110$ の場合も同様である．

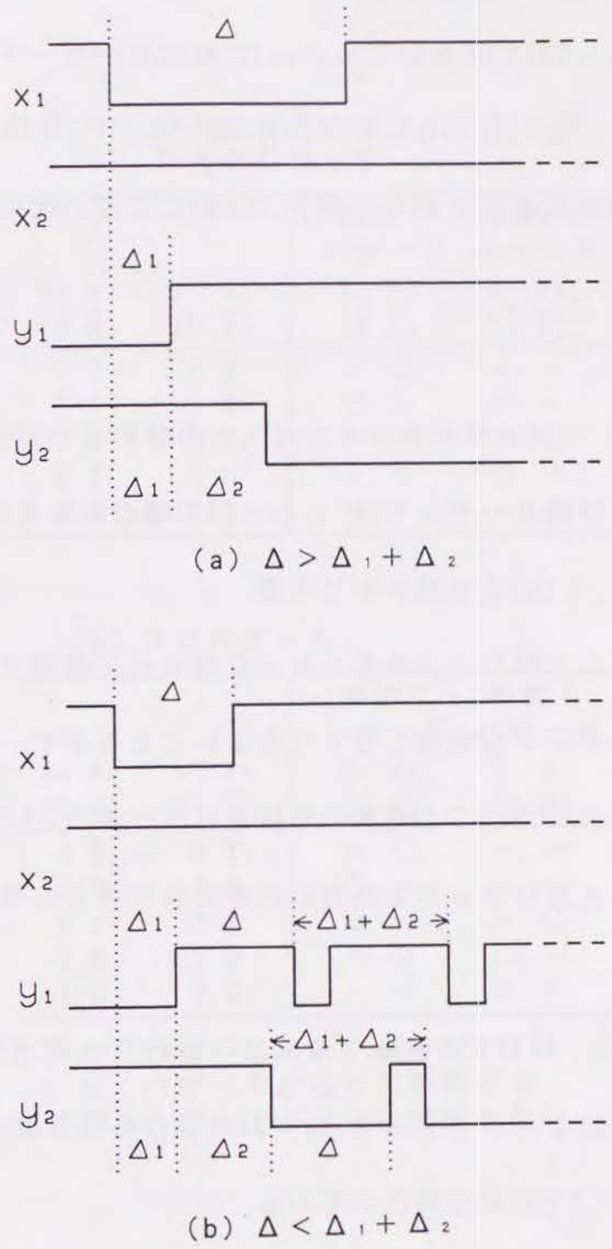


図２．６ タイミング図
（x_1に静的１ハザードによるグリッチがあるとき x_2=1，パルス幅Δ）

(3) $x_1x_2y_1y_2=0011$ において $x_1(x_2)$に静的0ハザードによるグリッチが入ると，$\Delta_1(\Delta_2)$後に $Y_1(Y_2)$に抜ける．

以上の解析結果をまとめると表2．1(a)になり，次の定理を得る[3]．

[定理2．1] ＮＡＮＤラッチでは入力の静的1ハザードは出力に影響するが，静的0ハザードは $x_1x_2=00$の場合を除き出力に出ない．

（証明）表2．1(a)より明らかである．

この定理はＮＡＮＤラッチをＳＲ＝0の条件で使用するとき，入力の静的0ハザードを考慮しなくてもよいことを示す．

ＮＯＲラッチについても同様の解析を行うと表2．1(b)を得る．この表からＮＡＮＤラッチと同様に次の定理が得られる[3]．

[定理2．2] ＮＯＲラッチでは入力の静的0ハザードは出力に影響するが，静的1ハザードは $x_1x_2=11$の場合を除き出力に出ない．

（証明）表2．1(b)より明らかである．

定理2．1と定理2．2のグリッチは，関数ハザードによるグリッチであってもよいことに注意すべきである．

表2．1　ＳＲラッチの出力とハザード

（a）ＮＡＮＤラッチ

		静的1ハザード	静的0ハザード
x_1 x_2 ($\bar{S}$ $\bar{R}$)	y_1 y_2 (Y $\bar{Y}$)	x_1 x_2 ($\bar{S}$ $\bar{R}$)	x_1 x_2 ($\bar{S}$ $\bar{R}$)
1 1	0 1	× ○	− −
1 1	1 0	○ ×	− −
1 0	0 1	× −	− ○
0 1	1 0	− ×	○ −
0 0	1 1	− −	× ×

（b）ＮＯＲラッチ

		静的1ハザード	静的0ハザード
x_1 x_2 (S R)	y_1 y_2 (Y $\bar{Y}$)	x_1 x_2 (S R)	x_1 x_2 (S R)
0 0	0 1	× ○	− −
0 0	1 0	○ ×	− −
0 1	0 1	× −	− ○
1 0	1 0	− ×	○ −
1 1	0 0	− −	× ×

注）− ：ハザードのない組合せ
　　× ：ハザードが出力に影響する
　　○ ：ハザードが出力に影響しない

2.4　SRラッチによる設計

順序回路の設計はふつう励起表から各状態変数と出力変数の論理式を求め図2.1(a)の形で実現される．非同期回路のとき論理式はハザードがないよう(ハザードフリー)に求めなければならない[1]．以下この方法を普通の設計法と呼ぶ．次にSRラッチを基本要素とする非同期回路（図2.1(b)）の設計法を述べる．順序回路の各状態変数にSRラッチ1つを割り当てる．このSRラッチの入力は順序回路の励起表に図2.2(d)のSRラッチの励起表を適用して得られる．すなわち，現在の状態 y_i と次の状態 Y_i の関係を図2.2(d)を使って S_i と R_i で表わすことができる．こうして得られたS,Rの入力を表わす表をSRの制御表[3]と呼ぶ（表2.2参照）．SRの制御表からラッチのS,R入力は積和形で

$$S = u_1 + u_2 + \cdots\cdots + u_n$$

$$R = v_1 + v_2 + \cdots\cdots + v_m \qquad (2.8)$$

の形に表わすことができ，$u_i(1 \leqq i \leqq n)$，$v_j(1 \leqq j \leqq m)$は入力変数と状態変数から成る積項である．こうして求めたS,Rをラッチの入力とする方法を以下SRラッチによる設計と呼ぶ[3]．

2.4.1　NANDラッチによる設計

基本モードで考えるとAND-OR（NAND2段）回路には静的1ハザードはあっても，静的0ハザードはない．したがって AND-OR-NOT 回路には静的0ハザードはあっても，静的1ハザードはない．(2.8)式をそのままNAND2段回路で構成すると図2.7(a)となる．このS,Rには静的0ハザードはなく，したがってS,Rには静的1ハザードはない．ふつう図2.7(a)の中間のNAND2段を束線化(bundling)[8]によって取り除き図2.7(b)のように設計される．

（補題2.1）　図2.7のSRラッチの入力には静的1ハザードはない．

（証明　略）

（補題2.2）　SRの制御表からS,Rを求めるとき，SR＝0は満たされているか，考慮する必要がない．

（証明）SRラッチの励起表（図2.2(d)）はyからYへのどの変化に対しても，必ず一方は0である．もとの順序回路にdon't careがなければ，SRの制御表でS,Rのいずれか一方は必ず0であり，SとRが最小項を共有することはない．もとの順序回路に don't care があると，それに対応するSRの制御表の最小項は共にdon't care になる．この最小項がSとRに共有され，SR＝0が満たされない可能性がある．しかし，don't care はもともと考慮する必要のない状態であるから，その状態になったときのハザードも考慮する必要はない．　（証明終）

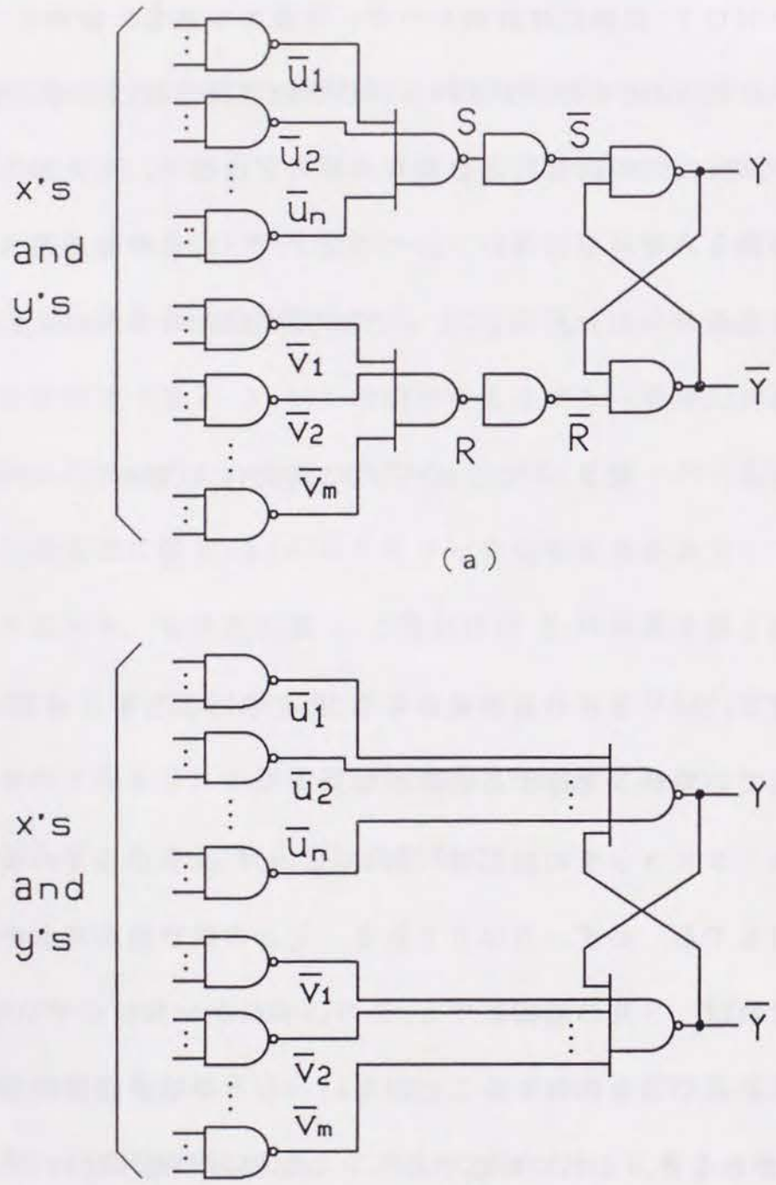


図２．７　ＮＡＮＤラッチによる構成

［定理２．３］　ＳＲラッチによる設計法でＮＡＮＤラッチを使って設計するとき，Ｓ，Ｒをハザードフリーに設計する必要はない[3].

（証明）補題２．１によりＳ，Ｒには静的１ハザードはなく，補題２．２によりＳＲ＝０を考慮する必要はない．したがって，定理２．１によりＳ，Ｒに静的０ハザードがあっても出力に影響しない．

（証明終）

２．４．２　ＮＯＲラッチによる設計

ＮＡＮＤラッチと同様に(2.8)式をＡＮＤ-ＯＲ２段回路で構成し，ＮＯＲラッチのＳ，Ｒ入力とすると図２．８(a)になる．このＳ，Ｒには静的０ハザードはない．またＳＲの制御表を用いて設計するとき，ＮＡＮＤラッチの場合と同様にＳＲ＝０は満たされるか，考慮する必要はない．

［定理２．４］　ＳＲラッチによる設計法でＮＯＲラッチを使って設計するとき，Ｓ，Ｒをハザードフリーに設計する必要はない[3].

（証明）Ｓ，Ｒには静的０ハザードはなく，前節の補題２．２よりＳＲ＝０は考慮する必要はない．よって定理２．２よりＳ，Ｒに静的１ハザードがあってもＮＯＲラッチの出力に影響しない．

（証明終）

図２．８(a)の回路はふつう途中のＯＲ回路を除き図２．８(b)また

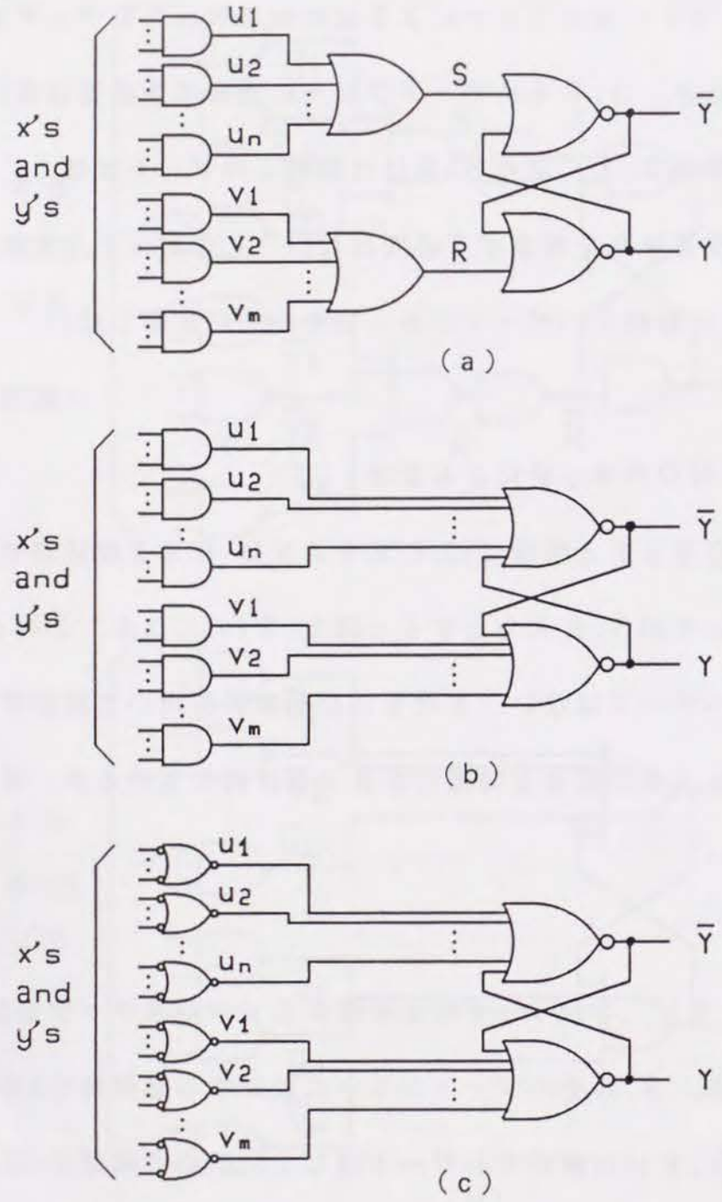


図２．８　ＮＯＲラッチによる構成

はＮＯＲ回路のみを使用して図２．８(c)のように構成される．

２．５　ＳＲラッチによる設計の特徴

まず，ＮＡＮＤラッチで構成された回路とＮＯＲラッチで構成された回路の間に次の定理が成立つ．

［定理２．５］　ＮＡＮＤラッチで構成された回路の入力のＮＡＮＤをＡＮＤに置き換え，出力の極性を逆にすれば，ＮＯＲラッチで構成できる．逆も同様である[3]．

（証明）(2.8)式がＮＡＮＤラッチでは図２．７(b)に，ＮＯＲラッチでは図２．８(b)になることから，入力を置き換えることは明らかである．また，ＳＲの制御表を使って設計されていれば，ＳＲ＝０を考慮せず，つねに出力の相補関係が保たれているので，出力の逆の極性は必ずとれる．　（証明終）

次に状態割当てについて考察する．一般にｕ状態にｒビット（ｒは $\log_2 u$ 以上の最小の整数）を割り当てる独立な割当て方は ${}_2{}^rP_u/(r!2^r)$ 通りある．この中には変数の０と１を入れ替えた割当ては独立とは考えていない．たとえば，４状態に２変数を割り当てるなら $4!/(2!2^2)=3$ 通りの独立な割当て方がある．同期回路では割当て方は任意であるが，非同期回路ではクリティカルレースがないよ

うに割り当てなければならない．どの割当てから最も簡単な回路が導かれるかは一般論はない．

状態変数 y_iの０と１を逆にすると，普通の設計法では Y_iの否定関数を求め y_iを $\overline{y_i}$としなければならない．こうすることによって回路の簡単さは異なる．しかし，ＳＲラッチによる設計法では単に y_iを $\overline{y_i}$として，S_iと R_iを入れ替えるだけであるから簡単さは変わらない．

普通の設計法でＹをハザードフリーに求めた式を(2.5)式の形に変形して設計することもできる．(2.5)式の形に変形し，ＮＡＮＤラッチのＳ，Ｒ入力を求めると，ＮＡＮＤラッチの出力としてＹを得ることができる．このときＳＲ＝０が成り立っていれば，① ＮＡＮＤラッチのもう一方の出力は$\overline{Y}$となり，② (2.7)式よりＮＯＲラッチでも構成でき出力はＹと$\overline{Y}$になる．しかし，一般にＳＲ＝０は成立つとは限らない．また，普通の設計法による構成とＳＲラッチによる設計とではどちらが簡単になるかは一般にわからない．

最後に回路の時間遅れを考える．入力変数の補元は得られるものとし，ゲートの遅れはすべて同じであると仮定する．普通の設計ではＹを２段回路で構成でき，Ｙの補元をとるためインバータが必要である（特殊な場合としてＹの補元を要しないこともある）．したがって，ふつう遅れは３ゲート分である．ＳＲラッチによる設計法

で構成したときは，ラッチの反転に２ゲート分必要であり，Ｓ，Ｒ入力は１段で構成できる．したがってどちらの設計法によっても最大時間遅れが３ゲート分になるように設計できる．

２．６　設計例

ＳＲラッチによる設計によって簡単な回路が得られる例を示す[3]．表２．２(a)の遷移表で示される順序回路を設計する．前節で述べたように状態変数 y_1, y_2の割当て方は３通りあるが，レースがないようにすると１通りになる（y_1, y_2の０，１を入れ替えた割当てはある）．この割当てを表２．２(a)の左端欄に示す．表２．２(b)から

$$\begin{aligned} Y_1 &= \overline{x_2} y_1 \overline{y_2} + x_1 \overline{x_2} y_1 + x_1 x_2 y_2 + \overline{x_1} x_2 y_1 \\ &\quad + \overline{x_1} y_1 \overline{y_2} + x_1 y_1 y_2 + x_2 y_1 y_2 \\ Y_2 &= \overline{x_1}\,\overline{x_2} y_2 + x_1 \overline{x_2}\,\overline{y_1} + x_1 x_2 y_2 + x_2 y_1 y_2 \\ &\quad + \overline{x_1} y_1 y_2 + x_1 \overline{y_1} y_2 + \overline{x_2}\,\overline{y_1} y_2 \end{aligned} \qquad (2.9)$$

となる．(2.9)式の $\overline{x_1} y_1 \overline{y_2}$，$x_1 y_1 y_2$，$x_2 y_1 y_2$，$\overline{x_1} y_1 y_2$，$x_1 \overline{y_1} y_2$，$\overline{x_2}\,\overline{y_1} y_2$はハザードフリーにするための項であり，$x_1 x_2 y_2$，$x_2 y_1 y_2$は$Y_1$と Y_2に共用できる．この設計では $\overline{y_1}$，$\overline{y_2}$のためにインバータが必要となり，回路の遅れは３ゲート分になる．ゲート数は１６，入力数は５２である．

つぎに表２．２(b)にＳＲラッチの励起表を用いて S_2R_2と S_1R_1の

表２．２　設計例

（a）遷移表

y_2y_1		x_2x_1 00	01	11	10
00	a	(a)	d	(a)	(a)
01	b	(b)	(b)	a	(b)
11	c	d	b	(c)	(c)
10	d	(d)	(d)	c	a

次の状態

（b）励起表

y_2y_1	x_2x_1 00	01	11	10
00	00	10	00	00
01	01	01	00	01
11	10	01	11	11
10	10	10	11	00

Y_2Y_1

（c）S_2R_2の制御表

y_2y_1	x_2x_1 00	01	11	10
00	0d	10	0d	0d
01	0d	0d	0d	0d
11	d0	01	d0	d0
10	d0	d0	d0	01

S_2R_2

（d）S_1R_1の制御表

y_2y_1	x_2x_1 00	01	11	10
00	0d	0d	0d	0d
01	d0	d0	01	d0
11	01	d0	d0	d0
10	0d	0d	10	0d

S_1R_1

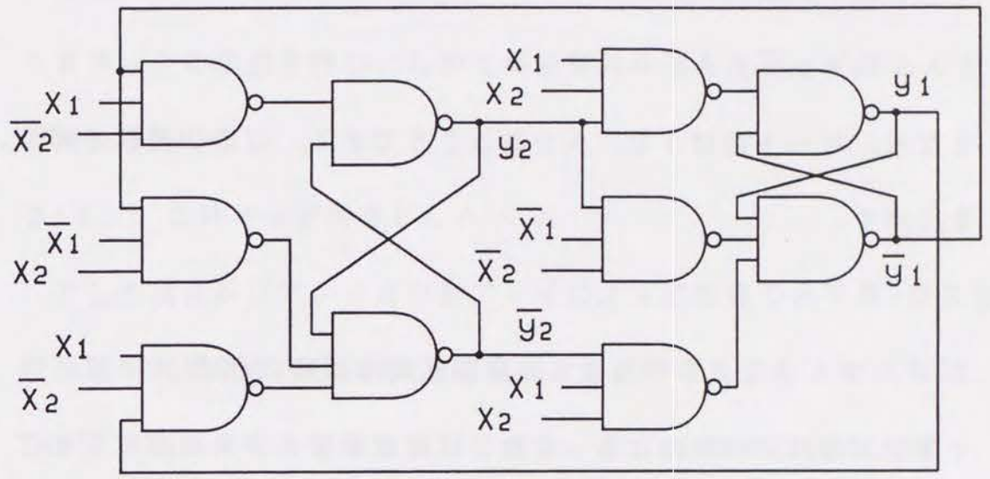


図２．９　設計例

制御表を作ると表２．２(ｃ)(ｄ)になる．これより

$$S_1 = x_1 x_2 y_2$$

$$R_1 = \overline{x_1}\,\overline{x_2} y_2 + x_1 x_2 \overline{y_2}$$

$$S_2 = x_1 \overline{x_2}\,\overline{y_1} \qquad (2.10)$$

$$R_2 = x_1 \overline{x_2} y_1 + \overline{x_1} x_2 \overline{y_1}$$

となり，ゲート数は１０，入力数は２８である．この回路図を図２．９に示す．

２．７　ＰＬＡとＲＯＭによる非同期式順序回路の合成

集積回路技術の発達によって論理回路はＰＬＡやＲＯＭなどのアレイロジックを用いて実現されるようになった．組合せ回路にはＡＮＤ-ＯＲ２段のＰＬＡやＲＯＭが，同期式順序回路にはレジスタ付きＰＬＡが広く使われている．非同期式順序回路を実現するにはレースやハザードの問題を解決しなければならないため，アレイロジックが用いられることは少ない．ＰＬＡを用いて非同期回路を実現する方法[9]やＲＯＭによてって実現する方法[10]が提案されているが，これらの文献では状態割当てによってレースを除去する方法が論じられている．

ここではアレイロジックで非同期式順序回路を合成するときのハザードの問題を考察する．ＡＮＤ-ＯＲ２段回路に存在する静的１ハザードはＳＲラッチによって吸収されるので，アレイの中にＳＲラッチを内蔵した型のＰＬＡを考えると，２．５で述べた方法で設計した回路をそのまま実現できる．このＰＬＡはＯＲアレイの出力部にたすきがけ回路を入れることによって，外部に付加回路を付けずにＳＲラッチの機能を持つ．しかも従来型のＰＬＡと比べてほとんど面積は増加しない．

２．７．１　ＳＲラッチ付きＰＬＡ

ＰＬＡはＡＮＤアレイとＯＲアレイによって任意のＡＮＤ-ＯＲ２段回路を実現する．(2.3)式を積和形で表現するとＰＬＡのＡＮＤ-ＯＲ２段回路でこの関数を実現できる．出力 Y_i をＰＬＡの入力 y_i にフィードバックすることによって(2.3)式で表わされる順序回路をＰＬＡで実現できる（図２．１１(a)参照）．以下このＡＮＤ-ＯＲ２段のみのＰＬＡを組合せ回路ＰＬＡと呼ぶ．

非同期式順序回路では静的１ハザードが存在するので，これを除去するために冗長な積項を追加しなければならない．積項の追加によってＰＬＡの積項線が増加し，必要なＰＬＡの面積が増大する．また，同期回路の合成のために開発された多数の優れた論理圧縮法を利用できない．

図２．７(a)の回路は組合せ回路ＰＬＡと外付けのＳＲラッチによって実現できるが，回路が複雑になり，段数が多いので遅れも大き

くなる．この欠点を除き，ＰＬＡで図２．７(b)の回路を実現する方法を示す．

図２．７(b)に示された回路はＮＡＮＤ２段回路の出力段にたすきがけ回路を追加した回路になっている．組合せ回路ＰＬＡのＯＲアレイの出力線を２本ずつ対にして一方の出力を他方の入力にゲートを通して接続することによって，図２．７(b)のたすきがけの部分を実現する．このたすきがけ回路がＳＲラッチの機能を持つ．以下このたすきがけ回路を内蔵したＰＬＡをＳＲラッチ付きＰＬＡと呼ぶ．

ＭＯＳによるＳＲラッチ付きＰＬＡの回路図を図２．１０に示す．このＡＮＤアレイとＯＲアレイは正論理ではＮＯＲ２段回路であるから，負論理でＮＡＮＤ２段回路になる．点線で囲まれた部分がたすきがけ回路を構成し，ＳＲラッチの機能を持つので，図２．１０のＰＬＡによって図２．７(b)の回路がそのまま実現される．また，ＳＲラッチのための回路に必要な面積はごく小量で，全体からみると無視できる程度である．

図２．１０の各交点の四角印はプログラム可能な点を示す．たすきがけ回路を構成するゲートもプログラム可能にされている．この部分の交点を切断すれば，たすきがけ回路はなくなり，ＳＲラッチの機能もなくなるので，組合せ回路ＰＬＡになる．

例として前節の設計例を２種類のＰＬＡを用いて実現した回路を

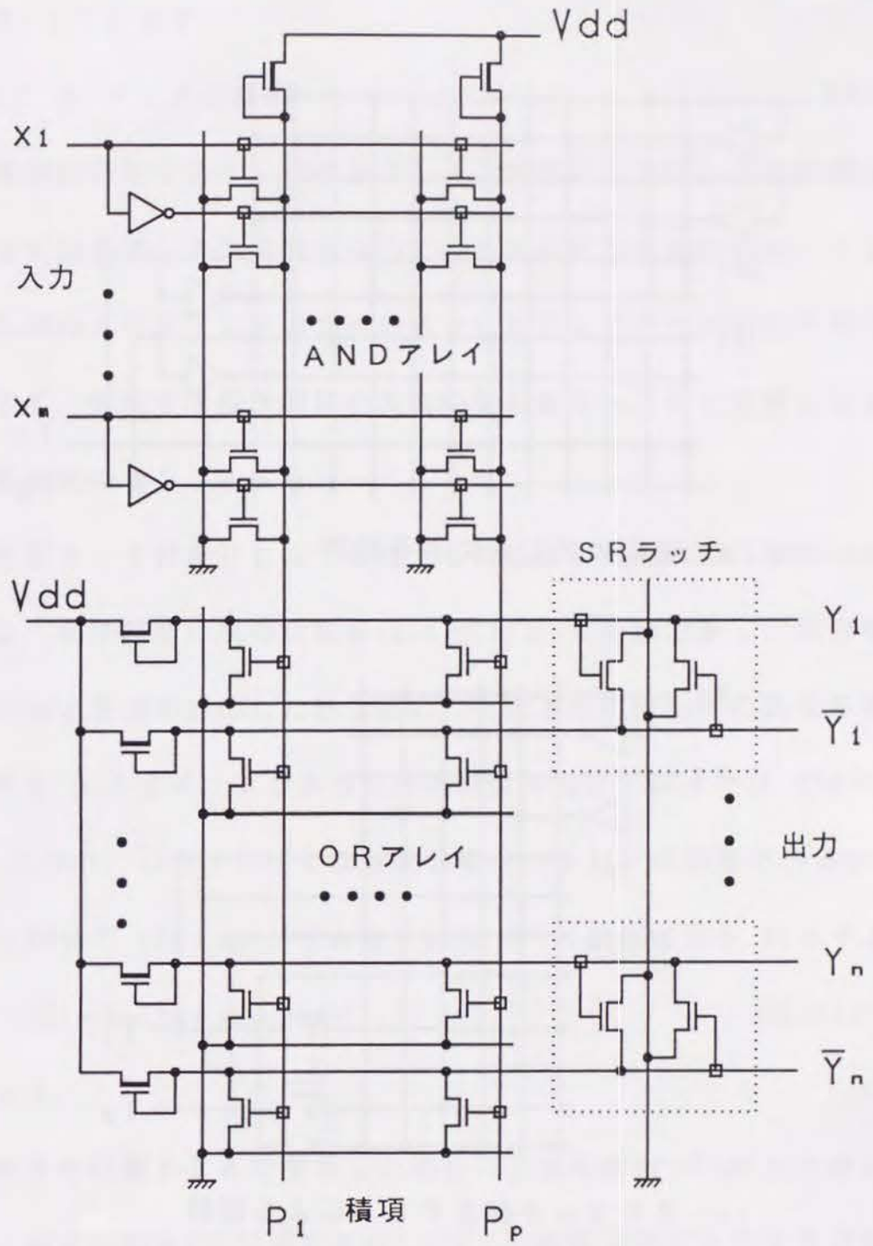
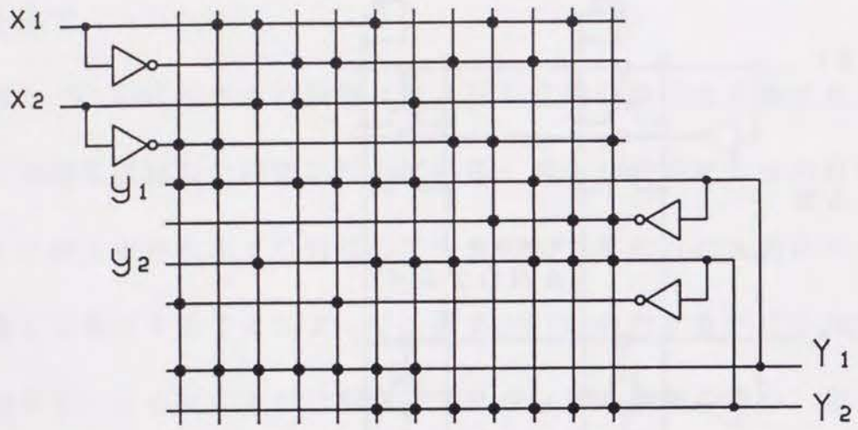
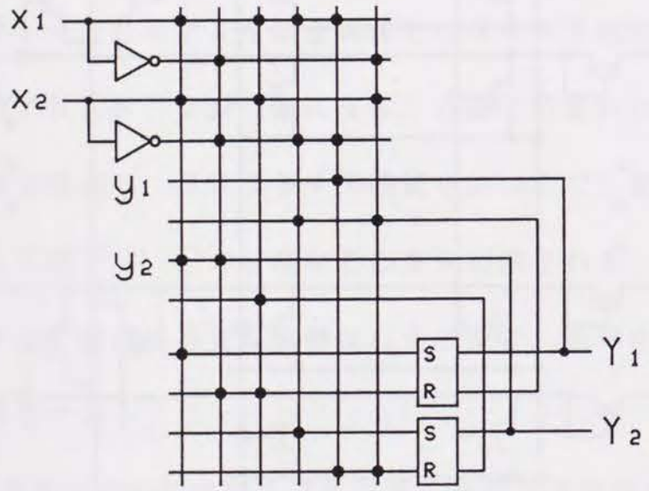


図２．１０　ＳＲラッチ付きＰＬＡ



(a) 従来のＰＬＡによる設計



(b) ＳＲラッチ付きＰＬＡによる設計

図２．１１ 設計例

図２．１１に示す．

２．７．２ ＰＬＡの面積

順序回路をＳＲラッチ付きＰＬＡで実現するのに必要な面積と，組合せ回路ＰＬＡで実現するのに必要な面積とを比較する．ＰＬＡの面積はＡＮＤアレイの交点の数とＯＲアレイの交点の数の和に比例する．実現する順序回路の入力変数の数をｎ，出力変数の数をｍ，状態変数の数をｑとする．

ＳＲラッチ付きＰＬＡで実現するのに必要な面積 W_1は次の様になる．順序回路の次の状態を(2.8)式のＳ，Ｒの形で表し，出力を(2.3)式のｇを積和形にした形で表わしたときの回路全体の異なる積項の数を p_1とする．ＰＬＡの交点の数はＡＮＤアレイでは $(2m+2q)\cdot p_1$となり，ＯＲアレイでは各状態変数にＳとＲが必要であるから出力と併せて $(2q+n)\cdot p_1$である．面積 W_1は比例定数を k_1とすると

$$W_1 = k_1(2m+4q+n)p_1 \qquad (2.11)$$

となる．

組合せ回路ＰＬＡで実現した場合に必要な面積 W_2は次の様になる．順序回路を(2.3)式で表わし，f_i, g_jを積和形にしたときの回路全体の異なる積項の数を p_2とする．ＡＮＤアレイはＳＲラッチ付きＰＬＡと同様 $(2m+2q)\cdot p_2$であり，ＯＲアレイは状態変数と出力であるから $(q+n)\cdot p_2$となる．面積 W_2は比例定数を k_2とすると

$$W_2 = k_2(2m + 3q + n)p_2 \qquad (2.12)$$

となる.

図２．１０からわかるように，k_1と k_2はほとんど同じである．(2.11.)式と(2.12)式では設計法が異なるので，積項数 p_1と p_2が異なり，W_1とW_2の大小関係はわからない．また，どちらのＰＬＡで実現する方が有利かを予め知る方法は知られていない．

前節の例では，ｍ＝２，ｑ＝２，ｎ＝０，p_1＝６，p_2＝１２であるから，W_1＝７２，W_2＝１２０である．この場合はＳＲラッチ付きＰＬＡで実現すれば組合せ回路ＰＬＡより４０％少ない面積ですむ（図２．１１参照）．また，各種フリップフロップをＰＬＡで実現したときに必要な面積を表２．３に示す（設計法は次章を参照）．この結果からはＳＲラッチ付きＰＬＡの方が少ない面積で実現されている．

２．７．３　ＲＯＭによる非同期式順序回路の合成

本節ではＲＯＭを用いて非同期式順序回路を実現する方法を述べ，それに必要なＲＯＭの容量（総ビット数）を求める．ＲＯＭは最小項の論理和の形で組合せ回路を実現するので，論理の圧縮ができない点と，静的１ハザードを除去できない特性がある．

非同期式順序回路をＲＯＭとフィードバックループで実現すると図１．１(a)の組合せ回路をＲＯＭで置き換えた形になる．順序回路

表２．３　各種フリップフロップを実現するために必要なＰＬＡの面積

	ＳＲラッチ付きＰＬＡ(W_1)	組合せ回路ＰＬＡ(W_2)	W_1/W_2
設計例	７２	１２０	０．６０
Ｄフリップフロップ	４０	５０	０．８０
マスタスレーブ型ＪＫフリップフロップ	５６	８４	０．６７
エッジトリガ型ＪＫフリップフロップ	８４	８４	１．００
エッジセンシングマスタスレーブ型ＪＫフリップフロップ	１０８	１６５	０．６５
ダブルエッジトリガ型ＪＫフリップフロップ	１２６	１３２	０．９５

を(2.3)式の形で表わし，すべての論理式を最小項の論理和の形式に変形すると Y_i, z_j はＲＯＭで実現できる．Y_i をＲＯＭの入力 y_i にフィードバックすると順序回路になる．このとき(2.3)式の静的１ハザードは論理的に除去できない．

図２．１(a)の組合せ回路をＲＯＭに置き換えて順序回路を実現するのに必要なＲＯＭの容量を求める．入力変数，出力変数，状態変数の数を，ＰＬＡの場合と同様に，それぞれ m，n，q とする．ＲＯＭの出力は（m＋q)であるから，語長は（m＋q)ビットである．ＲＯＭの入力が（n＋q)ビットであるから語数は (2^{m+q})語である．よってこの方法で順序回路を実現するのに必要なＲＯＭの総ビット数 C_2 は

$$C_2 = 2^{m+q} \cdot (n+q) \qquad (2.13)$$

となる．

ＰＬＡと同様にＲＯＭにもＳＲラッチを組み込むことができる．ＲＯＭの語（ワード）線を２本対にしてそれぞれ一方の出力をゲートを通して他方へ入力すると，この部分がたすきがけ回路となり，ＳＲラッチの機能を持つ．この型のＲＯＭはＳＲラッチによる設計法で設計した回路を直接実現できる．(2.8)式に静的１ハザードがあってもＳＲラッチに吸収される．しかし，出力はＳＲラッチを通らないのでハザードは残る．

ＳＲラッチ付きＲＯＭで順序回路を実現するのに必要なビット数を求める．ＲＯＭのワード線は各状態変数にＳとＲの２本必要であり，順序回路の出力と合わせて（n＋2q)となるので語長は（n＋2q)ビットである．語数は前節と同様に１語 (2^{m+q})である．必要なＲＯＭの容量 C_1 は

$$C_1 = 2^{m+q} \cdot (n+2q) \qquad (2.14)$$

ビットとなる．

ＲＯＭではＰＬＡのような論理の圧縮はできないので，ＲＯＭの容量に積項数は関係しない．したがって(2.13)式と(2.14)式を比較すると，つねに C_1 の方が C_2 より $(2^{m+q}) \cdot q$ ビット多く必要になる（語数は同じで語長が qビット長くなる)．ＳＲラッチ付きＲＯＭを用いると，必要なビット数の点では普通のＲＯＭを使うよりつねに不利であるが，状態変数のハザードを物理的に除去しなくてもよい利点がある．

２．８　おわりに

ＳＲラッチによる非同期式順序回路の設計法を述べた．ＳＲラッチのＳ，Ｒ入力にハザードがある場合を検討し，静的ハザードの極性によって出力に影響する場合としない場合を明確にした．出力に影響を及ぼさない静的ハザードしか論理的に存在しない２段回路でＳ，

R入力を構成すると，非同期回路を同期回路と同様の方法で設計できることを示した．

SRラッチによる設計法で設計した回路をそのまま実現できるPLAとして，SRラッチ付きPLAを示した．このPLAは組合せ回路PLAにごく少量の回路を追加するだけで構成でき，プログラムによってこの部分を削除することもできる．また，ROMでは静的1ハザードを論理的に除去することはできないが，SRラッチ付きROMを使用するとこのハザードを除去する必要がなくなる．しかし，必要なビット数は増加する．

参考文献

[1] S.H.Unger: Asynchronous Sequential Switching Circuits, Wiley-Inter-Science, New York(1969).

[2] D.B.Armstrong, A.D.Friedman and P.R.Menon: Realization of Asynchronous Sequential Circuits Without Inserted Delay Elements, IEEE Transactions on Computers, Vol.C-17, No.2, pp.129-134(1968).

[3] 井上訓行，奥川峻史：SRラッチによる非同期回路の設計，情報処理学会論文誌，Vol.28, No.10, pp1051-1059(1987).

[4] 井上訓行，奥川峻史：非同期回路理論によるラッチとフリップフロップの設計，電子通信学会技術研究報告，CPSY86-58(1987).

[5] 奥川峻史：LSIによる論理設計，共立出版(1987).

[6] J.Calvo, J.I.Acha and M.Valencia: Asynchronous Modular Arbiter, IEEE Transactions on Computers, Vol.C-35, No.1, pp.67-70(1986).

[7] 岡本卓爾，阿部山友邦：NANDラッチを対象としたメタステーブル動作の評価”，電子通信学会論文誌（D），Vol.J68-D, No.6, pp.1210-1217(1985).

[8] G.A.Maley and J.Earle: The Logic Design of Transistor Digital Computers, Prentice-Hall, Englewood Cliffs(1963).

[9] J.I.Acha, and J.Calvo: On the Implementation of Sequential Circuits with PLA Modules, IEE Proceedings Part E, Vol.135 No.5, pp.246-250(1985).

[10] A.S.Howard and S.Yang:Design of Asynchronous Sequential Networks Using READ-ONLY Menories, IEEE Transactions on Computers, Vol.C-24, No.2, pp.197-206(1975).

第３章　非同期回路理論に基づくフリップフロップの設計

３．１　はじめに

同期回路の基本構成要素であるラッチとフリップフロップは，それ自体はラッチとフリップフロップの入力とクロックを入力とする非同期回路である．これらの回路は多数知られており，その動作は説明されているが，同期回路の構成要素として扱われるため，回路そのものの設計について述べたものは少ない．すでに利用できるものとしてその設計について特に注意が払われない場合が多い．非常に基本的な問題であるが，これらの回路を非同期回路理論に基づいて理論的，系統的に設計することによって，その違いを明確にする．

まずはじめに，ゲート付きラッチの設計を示す．１つの遷移表からＥａｒｌｅラッチとＤラッチが導かれ，さらにエッジトリガ型Ｄフリップフロップの入力段になっている回路が導かれる[1]．この３つが本質的に同一のものであることが明らかにされる．

つぎに，Ｄフリップフロップではマスタスレーブ型とエッジトリガ型が１つの遷移表から導かれる[2]．その理由を明らかにする．

ＪＫフリップフロップでは，広く使われているエッジトリガ型ＪＫフリップフロップ３種類と典型的なマスタスレーブ型ＪＫフリップフロップが導かれる[1][3]．さらに，広く知られている回路の中

にいずれの定義からも導けない回路がある．クロックが１の間にＪ，Ｋ入力を変化させてはならない型を導入して，１つの遷移表から一般にエッジトリガ型と言われている回路とマスタスレーブ型と呼ばれている回路が導かれる．この２種類は状態遷移に関しては全く同じであって，出力のタイミングが異なるだけであることを示す[1]．

最後に特殊なフリップフロップとしてエッジセンシングマスタスレーブ型ＪＫフリップフロップ[3][4]とダブルエッジトリガ型ＪＫフリップフロップ[4][5]が導かれる．これらの設計において前章で述べたＳＲラッチによる設計法[4]の有効性が示される．

３．２　Ｄラッチの設計

Ｄラッチ（または単にラッチ）の動作は状態（出力）をＱ，クロック（またはイネーブル）入力をＣ，データ入力をＤとすると次のようになる．　① ＣがｌのときＱはＤに追従し（Ｑ＝Ｄ），　② Ｃが１から０に変化するとき（立下り）の状態を保存する．　③ Ｃが０のときＱは入力と切り離され変化しない．

Ｄラッチのシーケンス図（一部）を図３．１に，遷移表を表３．１(a)に示す．この遷移表は６状態である（Ｃ＝１，Ｑ≠Ｄの２状態はない）．表３．１(a)の状態を最小化すると表３．１(b)となり，状態変数ｙと出力が一致するように変数を割当てると表３．１(c)の励起

表を得る．この表から，ｙの次の値をＹとすると

$$Y = y\overline{C} + CD + yD^{*} \qquad (3.1)$$

となる．ｙＤはハザードフリーにするための項である．以下このための項も特に区別はしない

(3.1)式をそのままＮＡＮＤ回路で構成すると，Ｅａｒｌｅラッチ（図３．２(a)）[6]になる．ＥａｒｌｅラッチにはＡＮＤ-ＯＲ２段の組合せ回路を遅れなしに組み込める特徴があり，組合せ回路とラッチを交互に接続する高速のパイプライン処理装置に使用される[6]．(3.1)式を

$$Y = CD + (\overline{C} + D)y \qquad (3.2)$$

と変形し，(2.7)式によりＳ，Ｒを求めると

$$S = CD, \qquad R = C\overline{D} \qquad (3.3)$$

となる．ＮＡＮＤ回路で構成すると古典的なＤラッチ（図３．２(b)）となり，ＳＲ＝０であるから出力はＱ（Ｙ）と$\overline{Q}$である．なお，この回路はラッチの定義から直接導くこともできる．

つぎに，フリップフロップの入力段になっている回路を導く．(3.1)式を

$$Y = D\overline{\overline{C}\,\overline{y}} + \overline{C}\,y \qquad (3.4)$$

と変形し，ＮＡＮＤ３段回路で構成する（図３．３(a)）．$\overline{y}$のためのインバータを束線化(bundling)[7]により取り除きゲートの配置を

変えると図３．３(b)を得る．さらに，対称回路を求めるため図３．３(b)に示すように２つのＮＡＮＤの出力を Y_1，Y_2とすると

$$Y_1 = C + y_1\overline{y_2} + y_1\overline{D}$$

$$Y_2 = C + \overline{y_1} + y_2 D \qquad (3.5)$$

となる．$\overline{y_1}\,\overline{y_2} = \overline{C}\,y \cdot \overline{C}\,\overline{y} = 0$を用いて$Y_1$を変形すると

$$Y_1 = C + \overline{y_2} + y_1\overline{D} \qquad (3.6)$$

となり，これと(3.5)式の Y_2より図３．３(c)の対称回路を構成できる．図３．３のラッチは単独では使われていないが，後述のＤフリップフロップやＪＫフリップフロップの入力段に使われている．

図３．２の２つの回路と図３．３の３つの回路は，１つの遷移表（表３．１）から導かれたので，すべて論理的に同じものである．

３．３　Ｄフリップフロップの設計

フリップフロップにはエッジトリガ型とマスタスレーブ型がある．ポジティブエッジトリガ型Ｄフリップフロップはクロックｃの立上りで入力Ｄを取り込み，出力Ｑが変化する．Ｃがこれ以外のときＱは変化しない．

ポジティブエッジトリガ型Ｄフリップフロップのシーケンス図（一部）を図３．４に，遷移表を表３．２(a)に示す．この遷移表の状態を最小化すると４状態になり表３．２(b)と(c)となる．表３．３(b)

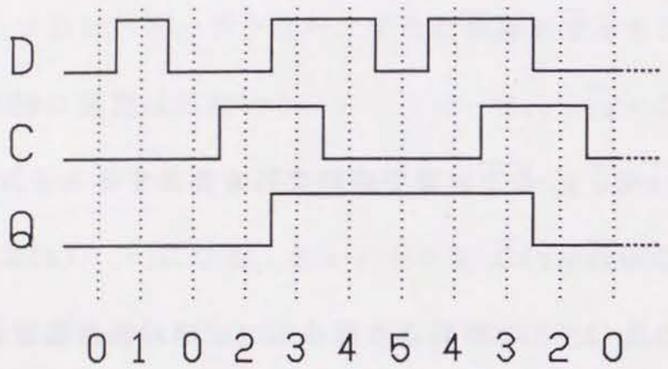


図3．1 ラッチのシーケンス図

表3．1 Dラッチ

（a）遷移表

現在の状態	CD 00	01	11	10
0	⓪，0	1，0	−，−	2，0
1	0，0	①，0	3，−	−，−
2	0，0	−，−	3，−	②，0
3	−，−	4，1	③，1	2，−
4	5，1	④，1	3，1	−，−
5	⑤，1	4，1	−，−	2，−

次の状態，出力

（b） 簡単化された遷移表

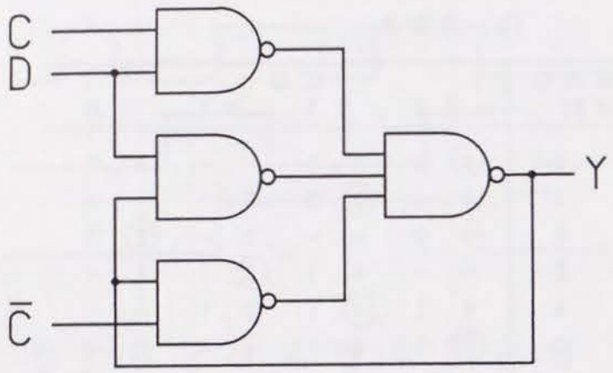
現在の状態	CD 00	01	11	10
(012) A	Ⓐ，0	Ⓐ，0	B，−	Ⓐ，0
(345) B	Ⓑ，1	Ⓑ，1	Ⓑ，1	A，−

次の状態，出力

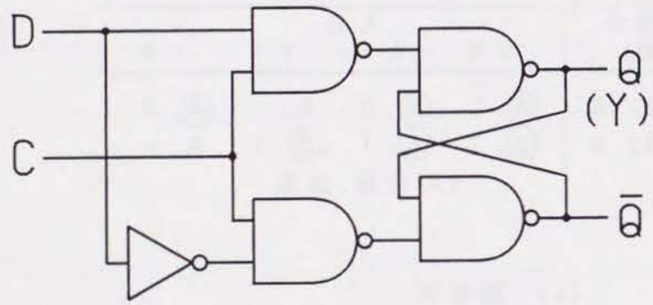
（c） 励起表

Y	CD 00	01	11	10
0	0	0	1	0
1	1	1	1	0

Y

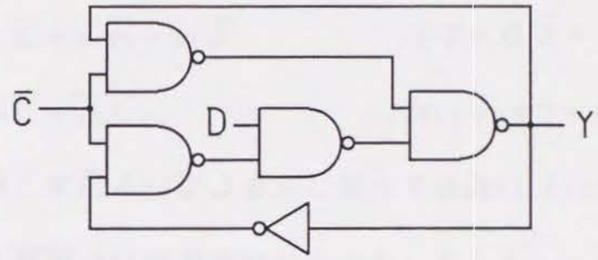


(a) Earle ラッチ

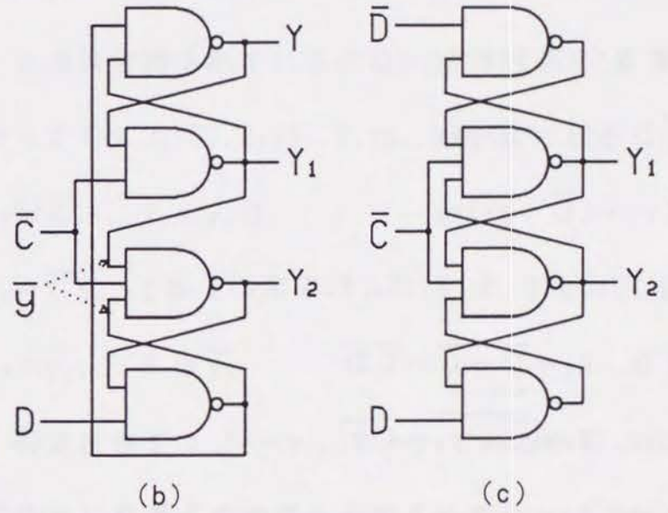


(b) Dラッチ

図3.2 Earle ラッチとDラッチ



(a)



(b) (c)

図3.3 各種のラッチ

にレースがないように，かつ y_2が出力と一致するように状態変数 y_1, y_2を割り当てると表３．２(d)を得る．この表より

$$Y_1 = C y_1 + \overline{C} D + D y_1$$

$$Y_2 = \overline{C} y_2 + C y_1 + y_1 y_2 \qquad (3.7)$$

となる．Y_1, Y_2は(3.1)式のＹと同じ形をしているので，$C y_1$を共通に使用するとＥａｒｌｅラッチを２つ縦続接続した回路となる（図３．５(a)）．y_1, y_2に２つのＳＲラッチを割り当て，その入力を S_1, R_1；S_2, R_2とする．(3.7)式を

$$Y_1 = \overline{C} D + (C + D) y_1$$

$$Y_2 = C y_1 + (\overline{C} + y_1) y_2 \qquad (3.8)$$

と変形し，(2.7)式より $S_1, R_1; S_2, R_2$を求めると

$$S_1 = \overline{C} D, \quad R_1 = \overline{D + C} = \overline{C}\,\overline{D}$$

$$S_2 = C y_1, \quad R_2 = \overline{\overline{C} + y_1} = C \overline{y_1} \qquad (3.9)$$

となる．$S_1 R_1 = S_2 R_2 = 0$であるから，２つのＳＲラッチの出力は Y_1と$\overline{Y_1}$, Y_2と$\overline{Y_2}$である．(3.9)式をＮＡＮＤラッチで構成すると図３．５(b)となる．これはマスタスレーブ型のＤフリップフロップである．

つぎに，広く使用されているエッジトリガ型Ｄフリップフロップ（図３．６(b)）を導く．この回路はＳＲラッチ３個からなっているので，状態変数３で設計する．表３．２(c)の４状態に状態変数 y_1, y_2, y_3を Ａ＝110，Ｂ＝111，Ｃ＝100，Ｄ＝011と割り当てる．レー

スが起こらないように don't care の代わりに必要な値を入れると，表３．２(e)が得られる．これより

$$Y_1 = \overline{C} + y_1 \overline{y_2} + y_1 \overline{D}$$

$$= \overline{C} + \overline{y_2 D}\, y_1$$

$$Y_2 = \overline{C} + \overline{y_1} + y_2 D \qquad (3.10)$$

$$= \overline{C y_1} + D y_2$$

$$Y_3 = \overline{y_1} + y_2 y_3$$

となる．出力は省略されているがＱ＝y_3である．(3.10)式から３つのＳＲラッチの入力 $S_1, R_1; S_2, R_2; S_3, R_3$を求めると

$$S_1 = \overline{C}, \quad R_1 = y_2 D$$

$$S_2 = \overline{C y_1}, \quad R_2 = \overline{D} \qquad (3.11)$$

$$S_3 = \overline{y_1}, \quad R_3 = \overline{y_2}$$

となる．安定状態では $Y_1 = y_1$，$Y_2 = y_2$であるから $S_3 R_3 = \overline{Y_1}\,\overline{Y_2} = 0$である．ＮＡＮＤラッチで構成すると図３．６(b)となる（図中のY_1，Y_2とは異なる）．

しかし，このＤフリップフロップはつぎに示すように状態変数２つで設計できる[2]．(3.7)式より

$$Y_1 = C y_1 + D(\overline{C} + y_1)$$

$$Y_2 = C y_1 + (\overline{C} + y_1) y_2 \qquad (3.12)$$

となる．ＮＡＮＤゲートで構成すると図３．６(a)となり，この図の

インバータを束線化によって取り除き，ゲートの配置を変えると図3．6(b)が得られる．さらに，図3．6(b)の入力側の4ゲートは前節の図3．3(b)と同じであるので，この部分を図3．3(c)で置き換えると図3．6(c)の対称回路となる．また，図3．6(c)のゲートの配置を変えると図3．6(d)が得られる．

エッジトリガ型Dフリップフロップの遷移表から，図3．5と図3．6のすべてのフリップフロップが設計された．図3．5(b)は典型的なマスタスレーブ型であり，図3．6(b)は典型的なエッジトリガ型である．したがって，Dフリップフロップでは，この2つは本質的に同じものである．JKフリップフロップ（次節参照）とは異なり，2つの型が同じになる理由はマスタスレーブ型でマスタの入力がDと$\overline{D}$であり，$\overline{C}$が1のとき（ポジティブエッジトリガの遷移表から導いたため普通のマスタスレーブ型とは異なりCが$\overline{C}$となっている）Dが変化すればマスタ側の出力はつねにDとなるように変化する．したがって，$\overline{C}$が1になる直前のDの値がスレーブに入り，出力となるからである．即ち，マスタスレーブ型Dフリップフロップはネガティブエッジトリガ型Dフリップフロップである．

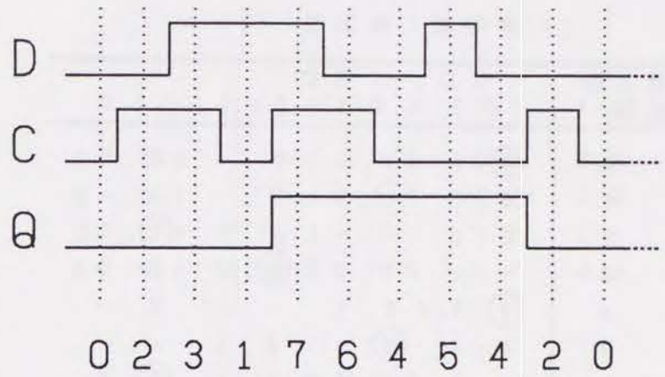


図3．4 Dフリップフロップのシーケンス図

表３．２　Ｄフリップフロップ

（ａ）遷移表

現在の状態	ＣＤ 00	01	11	10
0	⓪，0	1，0	－，－	2，0
1	0，0	①，0	7，－	－，－
2	0，0	－，－	3，0	②，0
3	－，－	1，0	③，0	2，0
4	④，1	5，1	－，－	2，－
5	4，1	⑤，1	7，1	－，－
6	4，1	－，－	7，1	⑥，1
7	－，－	5，1	⑦，1	6，1

次の状態，出力

（ｂ）簡単化された遷移表Ⅰ

現在の状態	ＣＤ 00	01	11	10
(1)　A	C，0	Ⓐ，0	D，－	－，－
(4)　B	Ⓑ，1	D，1	－，－	C，－
(023)　C	Ⓒ，0	A，0	Ⓒ，0	Ⓒ，0
(567)　D	B，1	Ⓓ，1	Ⓓ，1	Ⓓ，1

次の状態，出力

（ｃ）簡単化された遷移表Ⅱ

現在の状態	ＣＤ 00	01	11	10
(01)　A	Ⓐ，0	Ⓐ，0	D，－	C，0
(45)　B	Ⓑ，1	Ⓑ，1	D，1	C，－
(23)　C	A，0	A，0	Ⓒ，0	Ⓒ，0
(67)　D	B，1	B，1	Ⓓ，1	Ⓓ，1

次の状態，出力

表３．２　Ｄフリップフロップ（続き）

（ｄ）　励起表Ⅰ

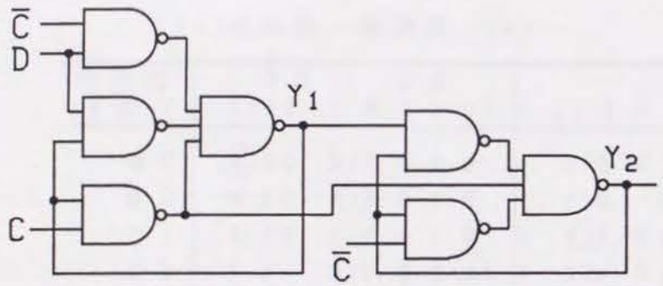
	y_1y_2	ＣＤ 00	01	11	10
C	00	00	10	00	00
B	01	01	11	dd	00
D	11	01	11	11	11
A	10	00	10	11	dd

Y_1Y_2

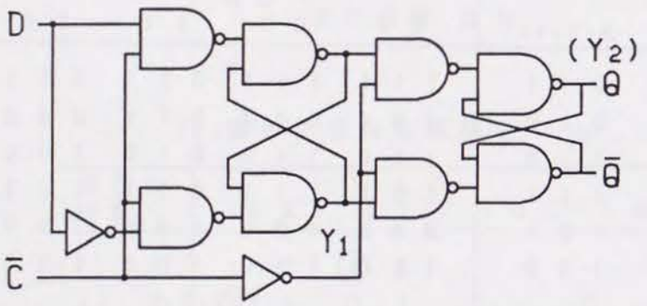
（ｅ）　励起表Ⅱ

	$y_1y_2y_3$	ＣＤ 00	01	11	10
D	011	111	111	011	011
	010	ddd	ddd	011	ddd
A	110	110	110	010	100
B	111	111	111	011	101
	101	ddd	ddd	ddd	100
C	100	110	110	100	100

$Y_1Y_2Y_3$

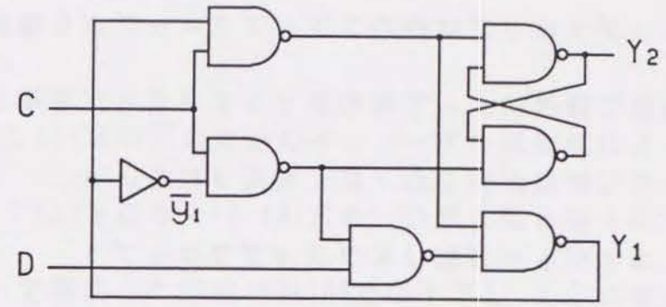


(a) Earle ラッチ2段による構成

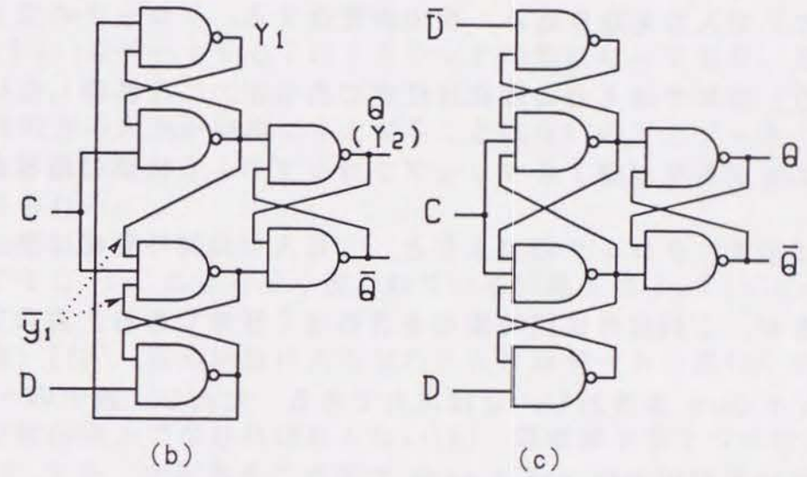


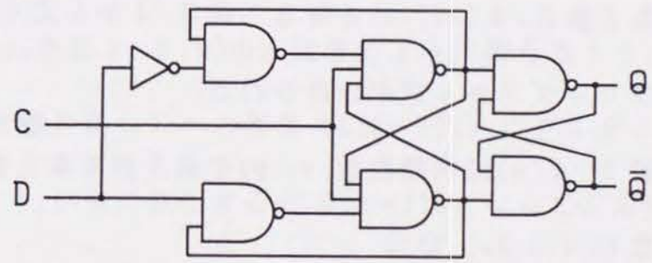
(b) マスタスレーブ型Dフリップフロップ

図3.5 Dフリップフロップ



(a)





(d)

図3.6 各種のDフリップフロップ

(注) (b)(c)が広く使われているエッジトリガ型
Dフリップフロップである.

3．4　ＪＫフリップフロップの設計

ＪＫフリップフロップは他のフリップフロップより機能的に優れており，簡単な操作によって他のフリップフロップを構成できるため，いろいろな回路が知られ，広く利用されている．

3．4．1　エッジトリガ型ＪＫフリップフロップ

エッジトリガ型ＪＫフリップフロップはクロックＣの立上り（立下り）で入力を取り込み，出力が変化する．クロックの立上り（立下り）以外では入力の変化は任意であり出力には影響しない．

エッジトリガ型ＪＫフリップフロップの１６状態の遷移表を表３．３に示す．クロックの立上りと Ｊ，Ｋ入力の同時変化は禁止されているが，これ以外は同時変化を含め全く任意である．表の空白は don't care を表わし，Ｑは出力である．ただし，表中の－印はその過渡状態の出力が don't care であることを示す．表３．３の状態を最小化すると表３．４(a),(b)を得る．表３．４から広く用いられている３つのフリップフロップが設計される．

まず，表３．４(a)に状態変数 y_1, y_2を割り当てると表３．５(a)の励起表になり Y_1，Y_2，Ｑは

$$Y_1 = y_1 C + y_1 \overline{y_2} J + y_1 y_2 \overline{K} + \overline{y_2}\,\overline{C}\, J + y_2 \overline{C}\,\overline{K}$$

$$= y_1 C + \overline{\overline{y_1} C} \cdot \overline{K} y_2 + \overline{\overline{y_1} C} \cdot J\, \overline{y_2 \overline{\overline{y_1} C}}$$

$$Y_2 = y_1 C + y_2 \overline{C} + y_1 y_2$$

$$= y_1 C + y_2 \overline{\overline{y_1} C} \qquad (3.13)$$

$$Q = y_2$$

となる．(3.13)式の$\overline{y_1}$のためのインバータを束線化によって取り除くと図３．７(a)を得る．(3.13)式のY_1の第２項と第３項をＡＮＤ-ＮＯＲゲートで構成した回路が74109型のＴＴＬ　ＩＣに使用されている[13]．

図３．７(a)のゲート６と７はＳＲラッチの形になっており，定常状態では両方の入力が同時に０となることはないので，ゲート７から$\overline{Q}$が得られる．

次にＴＴＬ　ＩＣにより広く使われている回路を図３．７(b)に示す（7473型）[13]．図の回路は入力段のＮＡＮＤゲートの遅れが他のゲートの数倍以上でなければならない[8]．遅延素子を１つの状態変数として非同期回路を設計することができる[9]．表３．４(a)の遷移表に y_1, y_2, y_3を表３．５(b)の左端欄のように割り当てる．この状態割当てでは状態ＢからＣへの遷移（010→111）にクリティカルレースが存在し，y_1が先に変化すると 010→110となって安定状態Ａになってしまう．ＢからＣへは 010→011→111と変化しなければならない．即ち y_1の変化を遅らせ y_3が先に変化しなければならない．励起表では過渡状態 011の入力 110,111の欄に 111を入れなければな

らない．DからAへの遷移も同様で y_3が y_2より先に変化し，101から過渡状態 100を経て 110となるようにしなければならない．以上のように過渡状態 011と 100を追加すると表3．5(b)の励起表となる．y_3が y_1, y_2より先に変化するように遅延素子を入れても表3．5(b)で他に不都合は生じない．表3．5(b)より

$$Y_1 = C + \overline{J} + y_2y_3 + \overline{C}y_3$$
$$= \overline{\overline{C}\ J\ y_2y_3 + \overline{C}y_3}$$

$$Y_2 = C + \overline{K} + \overline{y_3} \qquad (3.14)$$
$$= \overline{\overline{C}\,K\,y_3}$$

$$Y_3 = y_2y_3 + \overline{C}y_3 + C\overline{y_1}$$
$$= \overline{y_2y_3 + \overline{C}y_3\ (\overline{C} + y_1)}$$

$$Q = y_3$$

となり，図3．7(b)を得る．(3.14)式はハザードフリーにはなっていないが，図3．7(b)の y_1, y_2に遅延素子を入れなければならないので，誤動作は起こらない．(ＴＴＬ　ＩＣでは遅延素子に相当する遅れを持つゲートが使用されている．) $\overline{Q}$がQの補元出力になることも容易に確かめることができる．

最後に遅延を用いない対称回路を求める．表3．4(b)に y_1, y_2, y_3を表3．5(c)のように割り当てる．A(110)からD(101)への遷移にレースがあるため，100の入力 110, 111の欄に 101を入れなければ

ならない．このとき，A→C→Dの遷移を避けるため，AからDへは必らず 110→100→101となるように書き換える．CからBへの遷移についても同じである．表から

$$Y_1 = \overline{C} + \overline{y_2} + y_1\overline{y_3}\,J + y_3\overline{K}$$
$$= \overline{C} + \overline{y_2} + \overline{y_1y_3}y_1\,J + \overline{y_2y_3K}\,\overline{\overline{y_1y_3}y_1\,J}\,y_3$$

$$Y_2 = \overline{C} + \overline{y_1} + y_2y_3K + \overline{y_3}\,\overline{J}$$
$$= \overline{C} + \overline{y_1} + y_2y_3K + \overline{y_2y_3K}\,\overline{\overline{y_1y_3}y_1\,J}\,\overline{y_1y_3} \qquad (3.15)$$

$$Y_3 = \overline{y_2} + y_1y_3$$
$$= \overline{\overline{y_1y_3}y_2}$$

が得られ，図3．7(c)[10]に示す対称回路が設計できる．$Y_1Y_2 = 0$であるから，出力はQ(Y_3)と$\overline{Q}$が得られる．

以上に示したように図3．7はすべて論理的に等価な回路であり，ＴＴＬ　ＩＣでは図3．7(b)の回路が最も広く用いられている（クロックの極性は逆になっている）．回路構成が簡単で対称回路であるためと考えられる．

3．4．2　マスタスレーブ型ＪＫフリップフロップ

マスタスレーブ型ＪＫフリップフロップはクロックが1の間に，入力（Q＝0ならＪ＝1，Q＝1ならＫ＝1）がマスタフリップフロップに取り込まれ，次のクロックの立下りでスレーブフリップフロップに送られ出力が変化する．このような入力がなければ出力は

表３．３　エッジトリガ型ＪＫフリップフロップの遷移表

現在の状態	ＣＪＫ 000	001	011	010	110	111	101	100	出力Q
0	⓪	1	3	2				4	0
1	0	①	3	2			5		0
2	0	1	3	②	e-				0
3	0	1	③	2		f-			0
4	0	1	3	2	6	7	5	④	0
5	0	1	3	2	6	7	⑤	4	0
6	0	1	3	2	⑥	7	5	4	0
7	0	1	3	2	6	⑦	5	4	0
8	⑧	9	b	a				c	1
9	8	⑨	b	a			5-		1
a	8	9	b	ⓐ	e				1
b	8	9	ⓑ	a		7-			1
c	8	9	b	a	e	f	d	ⓒ	1
d	8	9	b	a	e	f	ⓓ	c	1
e	8	9	b	a	ⓔ	f	d	c	1
f	8	9	b	a	e	ⓕ	d	c	1

次の状態

（注）空欄は don't care を示す．また，状態の後の－はその状態の出力が don't care であることを示す．

表３．４　エッジトリガ型ＪＫフリップフロップの簡単化された遷移表

（a）

現在の状態	ＣＪＫ 000	001	011	010	110	111	101	100	出力Q
(0, 1, 4-7) A	Ⓐ	Ⓐ	B	B	Ⓐ	Ⓐ	Ⓐ	Ⓐ	0
(2, 3) B	A	A	Ⓑ	Ⓑ	C	C	-	-	0
(8, a, c-f) C	Ⓒ	D	D	Ⓒ	Ⓒ	Ⓒ	Ⓒ	Ⓒ	1
(9, b) D	C	Ⓓ	Ⓓ	C	-	A	A	-	1

次の状態

（b）

現在の状態	ＣＪＫ 000	001	011	010	110	111	101	100	出力Q
(0, 1, 2, 3) A	Ⓐ	Ⓐ	Ⓐ	Ⓐ	D	D	B	B	0
(4, 5, 6, 7) B	A	A	A	A	Ⓑ	Ⓑ	Ⓑ	Ⓑ	0
(8, 9, a, b) C	Ⓒ	Ⓒ	Ⓒ	Ⓒ	D	B	B	D	1
(c, d, e, f) D	C	C	C	C	Ⓓ	Ⓓ	Ⓓ	Ⓓ	1

次の状態

表３．５　エッジトリガ型ＪＫフリップフロップの励起表

(a)

	y_1y_2	C J K 000	001	011	010	110	111	101	100
A	00	00	00	10	10	00	00	00	00
B	10	00	00	10	10	11	11	dd	dd
C	11	11	01	01	11	11	11	11	11
D	01	11	01	01	11	dd	00	00	dd
		Y_1Y_2							

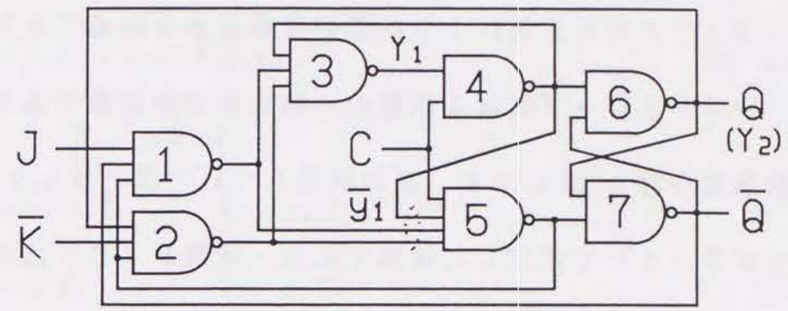
(b)

	$y_1y_2y_3$	C J K 000	001	011	010	110	111	101	100
	011	ddd	ddd	ddd	ddd	111	111	ddd	ddd
B	010	110	110	010	010	111	111	ddd	ddd
A	110	110	110	010	010	110	110	110	110
C	111	111	101	101	111	111	111	111	111
D	101	111	101	101	111	ddd	110	110	ddd
	100	ddd	ddd	ddd	ddd	ddd	110	110	ddd
		$Y_1Y_2Y_3$							

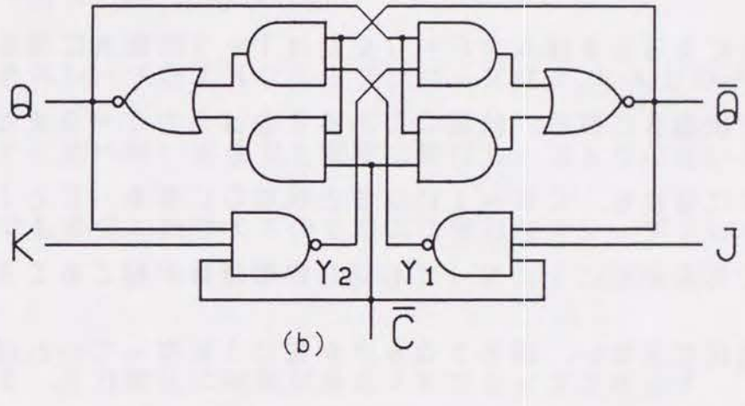
(c)

	$y_1y_2y_3$	C J K 000	001	011	010	110	111	101	100
	011	ddd	ddd	ddd	ddd	ddd	010	010	ddd
B	010	110	110	110	110	010	010	010	010
A	110	110	110	110	110	100	100	010	010
C	111	111	111	111	111	101	011	011	101
D	101	111	111	111	111	101	101	101	101
	100	ddd	ddd	ddd	ddd	101	101	ddd	ddd
		$Y_1Y_2Y_3$							

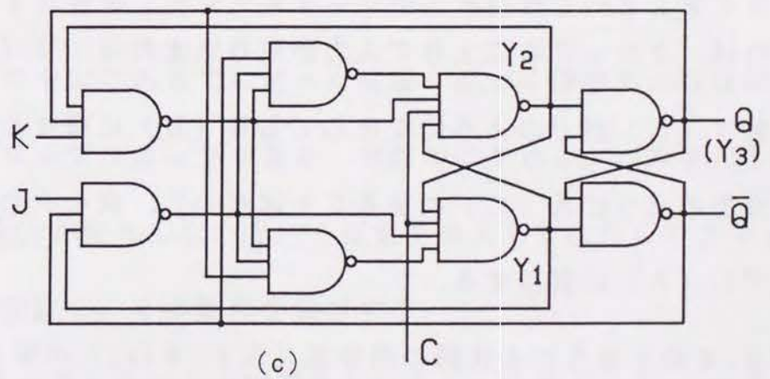
（注）　(b)(c)の 000,001 の行はすべて don't care であるから省略されている.



(a)



(b)



(c)

図３．７　エッジトリガ型ＪＫフリップフロップ

変化しない.

エッジトリガ型と同様に１６状態の遷移表から始めても設計できるが，マスタスレーブ型は４状態から始める方が簡単である．４状態の状態図を図３．８に示す．図の状態Ａ（Ｃ）はクロックＣが０，出力Ｑが０（１）で安定した状態である．状態Ｂ（Ｄ）はクロックが１，出力が０（１）で，次にクロックＣが１→０のときＣ（Ａ）に移る準備ができている状態である．

状態Ａにあるときは入力Ｃ＝０またはＪ＝０の間Ａに留まり，ＣＪ＝１で状態Ｂに移る．状態Ｃにあるときは入力Ｃ＝０またはＫ＝０の間Ｃに留まり，ＣＫ＝１になると状態Ｄに移る．ＣとＪ（ＣとＫ）のどちらが先に１になっても同じ状態遷移が起こることに注意しなければならない．即ちクロックが先に１になっていれば，Ｊ，Ｋ入力の最初の立上りで入力が取り込まれ，Ｊ，Ｋ入力が先に１になっていれば，クロックの立上りで入力が取り込まれる．Ｂ（Ｄ）に移った後Ｊ（Ｋ）がどのように変化してもＢ（Ｄ）に留まり（エッジトリガ型のようにＡ（Ｃ）に戻ることはない），次のクロックの立下りでＣ（Ａ）に遷移する．

図３．８から直ちに４状態の遷移表（表３．６(a)）が得られ，励起表は表３．６(b)となる．この表より

$$Y_1 = y_1\overline{C} + y_1\overline{y_2} + y_1\overline{K} + \overline{y_2}CJ$$

$$Y_2 = y_1\overline{C} + y_2C + y_1y_2 \qquad (3.16)$$

となり，$S_1, R_1; S_2, R_2$は

$$S_1 = \overline{y_2}CJ \qquad R_1 = y_2CK$$

$$S_2 = \overline{C}y_1 \qquad R_2 = \overline{y_1}\,\overline{C} \qquad (3.17)$$

となる[3]．(3.17)式から典型的なマスタスレーブ型ＪＫフリップフロップ（図３．９）が導かれる．

マスタスレーブ型ＪＫフリップフロップはＪ，Ｋ入力のパルス幅がクロックに比べ狭い場合でも安定に動作し，立上りの遅いクロックパルスでも確実に動作するので非常に便利なフリップフロップである[10]．

３．４．３　入力変化に制限があるＪＫフリップフロップ

エッジトリガ型，マスタスレーブ型と呼ばれているＪＫフリップフロップの中にこれまでに述べた定義に基づく遷移表からは設計できないフリップフロップがある．本節ではクロックパルスが１の間にＪ，Ｋ入力が変化しないという仮定を導入してこれらのフリップフロップを設計し，その動作を検討する．

表３．３の遷移表に上の制限を入れると，状態４～７，ｃ～ｆのクロックが１の部分は安定状態のみとなり，クロックが０の部分はそ

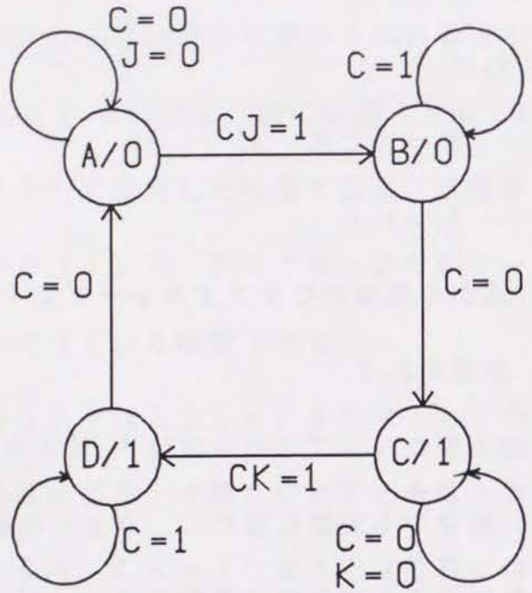


図３．８ マスタスレーブ型
ＪＫフリップフロップの状態図

表３．６　マスタスレーブ型ＪＫフリップフロップ

（a）遷移表

現在の状態	ＣＪＫ 000	001	011	010	110	111	101	100	出力 Q
A	(A)	(A)	(A)	(A)	B	B	(A)	(A)	0
B	C	C	C	C	(B)	(B)	(B)	(B)	0
C	(C)	(C)	(C)	(C)	(C)	D	D	(C)	1
D	A	A	A	A	(D)	(D)	(D)	(D)	1

次の状態

（b）励起表

y_1y_2	ＣＪＫ 000	001	011	010	110	111	101	100
A 00	00	00	00	00	10	10	00	00
B 10	11	11	11	11	10	10	10	10
C 11	11	11	11	11	11	01	01	11
D 01	00	00	00	00	01	01	01	01

Y_1Y_2

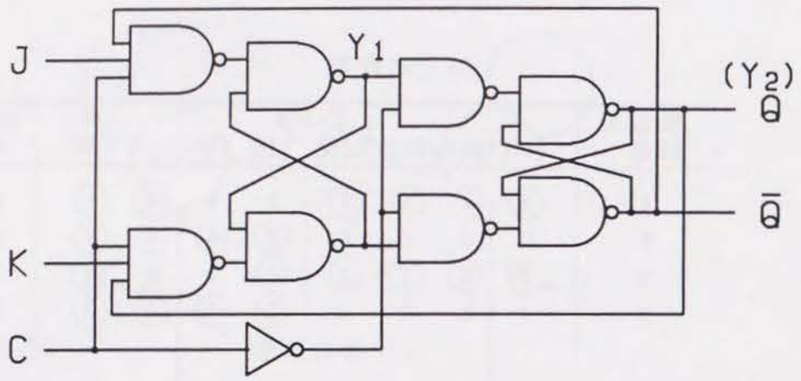


図３．９ マスタスレーブ型ＪＫフリップフロップ

れぞれの安定状態からクロックが１→０となるときの過渡状態だけになる．これに伴って状態６とｄへの遷移はなくなるのですべてdon't care にする．こうして得られる遷移表（表は省略）の状態を最小化すると４状態の遷移表（表３．７(a)）が得られ，レースがないように状態変数 y_1, y_2を割り当てると表３．７(b)の励起表になる．表より

$$Y_1 = y_1 C K + y_2(y_1 + \overline{C} + \overline{J})$$

$$Y_2 = \overline{y_1} C J + y_2(\overline{y_1} + \overline{C} + \overline{K}) \qquad (3.18)$$

となる．(3.16)式を

$$Y_1 = y_2\overline{\overline{y_1} C J} + y_1 C K$$

$$Y_2 = y_2\overline{y_1 C K} + \overline{y_1} C J \qquad (3.19)$$

と変形し，$\overline{y_1}$を束線化によって，インバータを使わずに構成すると図３．１０(a)となり，出力はＱ＝y_2である．出力段のＳＲラッチの入力は同時に０にはならないのでもう一方の出力は$\overline{Q}$となる．また，図３．３のラッチと同様の変形によって図３．１０(b)に示す対称回路も構成できる．(3.18)式を

$$Y_1 = y_2\overline{\overline{y_1} C J} + (y_2 + y_1 C K)y_1$$

$$Y_2 = \overline{y_1} C J + (\overline{y_1} + \overline{C} + \overline{K})y_2 \qquad (3.20)$$

と変形して，ＳＲラッチの入力を求めると

$$S_1 = y_2\overline{\overline{y_1}CJ}, \qquad R_1 = \overline{y_2}\,\overline{y_1CK}$$

$$S_2 = \overline{y_1}CJ, \qquad R_2 = y_1CK \qquad (3.21)$$

となり，ＳＲラッチを２個使った回路（図３．１０(c)）が得られる．図３．１０(c)の回路の出力はふつう $Q = y_1$となっている．２つのＳＲラッチの入力はいずれもＳＲ＝０を満たしているのでゲート８から$\overline{Y_1}(\overline{Q})$，ゲート４から $\overline{Y_2}$が得られる．

図３．１０の回路はすべて状態遷移に関して等価な回路であるが出力のタイミングが異なる．状態変数 y_2はクロックの立上りでＡ→Ｄ，Ｃ→Ｂの遷移によって変化し，y_1はクロックの立下りでＢ→Ａ，Ｄ→Ｃの遷移によって変化する．図３．１０(a),(b)の出力は $Q = y_2$であるからクロックの立上りで変化し，図３．１０(c)では $Q = y_1$であるからクロックの立下りで変化する．

一般に図３．１０(a),(b)はエッジトリガ型と呼ばれいる．この理由はクロックの立上りで入力を取り込み出力が変化するからと考えられる．しかし，図３．１０(a),(b)ではクロックが０になるまでに入力は変化してはならない．したがって３．４．１で設計したエッジトリガ型と等価ではない．クロックが１の間に入力が変化すると誤動作することは容易に確かめることができる．例えば図３．１０(a)で ＣＪＫy_1y_2＝10000 の安定状態にあるとき，Ｊが０→１と変化す

ると 11001となり Y_2は０→１と変わってしまう．入力変化に制限が付いただけ回路が簡単になっている．Ｄフリップフロップではこのようなことが起こらないので，図３．１０(a)と類似の回路が広く使われている．

図３．１０(c)はマスタスレーブ型と呼ばれている．クロックの立上りで入力を一段目（y_2）のラッチに取り込み，クロックの立下りで出力が変化する点では前節のマスタスレーブ型と同じであるが，入力変化にエッジトリガ型と同様の制限が付く．クロックが１になってからＪ（Ｋ）が１になっても正しい動作をするが，一度１になった入力がＣ＝１の間に０に戻ると誤動作をする．

ここで述べたフリップフロップは３．４．１および３．４．２に示したフリップフロップより回路構成が簡単であるから，使用する同期回路が入力変化の制限条件を満たすなら効果的に利用できる．しかし，入力変化に制限があることに注意する必要があり，図３．１０(a),(b)とエッジトリガ型，図３．１０(c)とマスタスレーブ型を全く同じ機能を持つと考えるべきではない．

３．４．４　本質的ハザード

ＪＫフリップフロップの遷移表にはすべて本質的ハザードが存在する[3]．図３．９の回路ではクロックを反転させるインバータが他のゲートに比べ３倍以上遅いとクロックの立下りで入力がマスタと

表３．７　入力変化に制限があるＪＫフリップフロップ

(a)遷移表

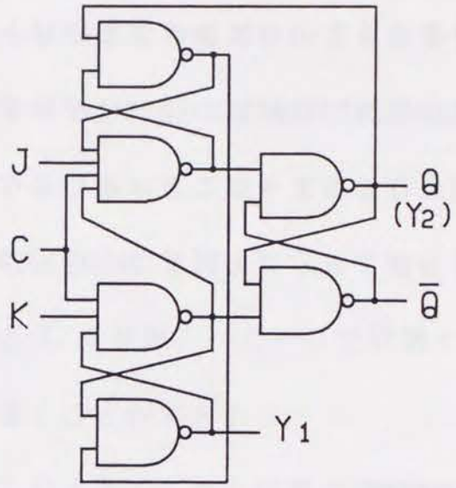
現在の状態	CJK 000	001	011	010	110	111	101	100
(0-6) A	(A)	(A)	(A)	(A)	D	D	(A)	(A)
(5,7) B	-	A	A	-	-	(B)	(B)	-
(8-e) C	(C)	(C)	(C)	(C)	(C)	B	B	(C)
(e,f) D	-	-	C	C	(D)	(D)	-	-

次の状態

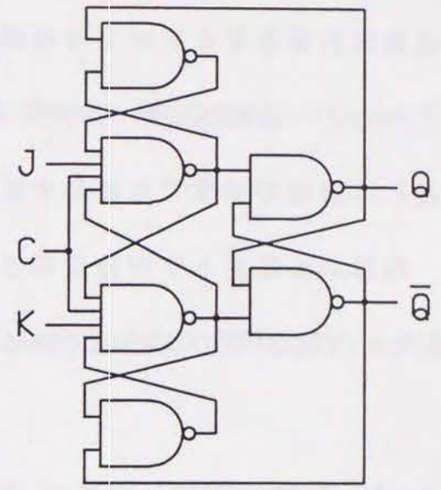
(b) 励起表

	y_1y_2	CJK 000	001	011	010	110	111	101	100
A	00	00	00	00	00	01	01	00	00
B	10	dd	00	00	dd	dd	10	10	dd
C	11	11	11	11	11	11	10	10	11
D	01	dd	dd	11	11	01	01	dd	dd

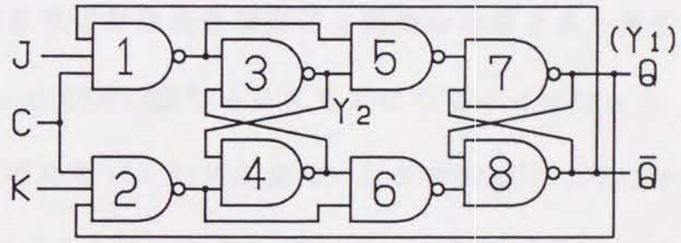
Y_1Y_2



(a)



(b)



(c)

図３．１０　各種のＪＫフリップフロップ

スレーブを素通りする．この程度の遅れのばらつきはそう希ではない[3]．しかし，図３．１０(c)ではゲート１の出力がゲート３の変化後の出力より遅くゲート５に達するときにのみ誤動作する．ゲート１から５へは配線だけであり，集積回路では配線の遅れはゲートの遅れに対して無視できるので現実にはこのようなことは起こらない．

なお，エッジトリガ型ＪＫフリップフロップと図３．１０(a),(b)についても同様の検討ができる．

３．５　特殊なフリップフロップの設計

本節では２種類の特殊なフリップフロップを設計し，普通の方法と第２章で述べたＳＲラッチによる設計を比較検討する．

３．５．１ エッジセンシングマスタスレーブ型ＪＫフリップフロップ

エッジセンシングマスタスレーブ型ＪＫフリップフロップはクロックの立上がりでＪ，Ｋ入力を取込み，クロックの立下がりで出力が変化する[3]．このフリップフロップの遷移表（表３．８(a)）は文献[3]の図１５．９に示されているので，ここでは概略を示す．

フリップフロップの出力は z_2であり，クロックＣが０のとき安定状態ａ(出力０)またはｃ(出力１)にある．状態ａでＣが０→１と変化するとき，Ｊ＝１ならｂに移る．ｂは次にクロックが０になると出力が変化して状態ｃに移る準備状態である．Ｊが０なら状態ｅに

移り，次にクロックが０になるとａに戻る．出力が変化しない入力による状態遷移である．状態ｂ，ｅともＣ＝１の間にＪ，Ｋが変化しても状態は変わらない．すなわちＣが０→１になるときのＪの値(Ｋは無関係)によって状態がｂかｅに遷移する．普通のマスタスレーブ型ＪＫフリップフロップではＣＪ＝１で状態ａからｂに移るが，Ｃ，Ｊのどちらが先に１になっても状態ｂに移るので，状態ｅはない（表３．６参照）．Ｃ＝０で状態ｃにあるときもＪをＫに変えて同様に導くことができる．

このようにした６状態の遷移表を表３．８(a)に示す．表３．８(a)の各状態に状態変数 y_1, y_2, y_3をレースがないように割り当てる（表３．８(b)の左端欄）．この割当ては変数の順序と０，１の入替えを除いてユニークである．

表３．８(b)から Y_1, Y_2, Y_3と出力 Z_1, Z_2をハザードフリーに求めると，

$$Y_1 = y_1y_3 + y_1\overline{y_2} + y_1\overline{C} + y_1\overline{K} + C\,J\,\overline{y_2}\,\overline{y_3}$$

$$Y_2 = \overline{C}\,y_1 + C\,y_2 + y_1y_2$$

$$Y_3 = C\,y_3 + C\,\overline{J}\,\overline{y_1}\,\overline{y_2} + C\,\overline{K}\,y_1y_2 \qquad (3.22)$$

$$Z_1 = y_1$$

$$Z_2 = y_2$$

となる[3]．y_1, y_2, y_3のためのインバータを含め１５ゲート，入力総

数４０である.

次に表３．８(b)から S_1R_1 の制御表を作ると表３．８(c)になる(S_2, R_2;S_3,R_3も同様). これから各S,Rを求めると,

$$S_1 = C\,J\,\overline{y_2}\,\overline{y_3} \qquad R_1 = C\,K\,y_2\,\overline{y_3}$$

$$S_2 = \overline{C}\,y_1 \qquad R_2 = \overline{C}\,\overline{y_1} \qquad (3.23)$$

$$S_3 = C\,\overline{J}\,\overline{y_1}\,\overline{y_2} + C\,\overline{K}\,y_1 y_2 \qquad R_3 = \overline{C}$$

となり, １２ゲート, 入力総数３３である（図３．１１）. したがって, この設計ではＳＲラッチによる設計が有効である.

(3.22)式を(2.5)式の形に変形すると

$$Y_1 = C\,J\,\overline{y_2}\,\overline{y_3} + (\overline{C} + \overline{K} + \overline{y_2} + y_3)\,y_1$$

$$Y_2 = \overline{C}\,y_1 + (C + y_1)\,y_2 \qquad (3.24)$$

$$Y_3 = C\,\overline{J}\,\overline{y_1}\,\overline{y_2} + C\,\overline{K}\,y_1 y_2 + C\,y_3$$

となり, これよりS,Rを求めると(3.23)式と同じになる. ($S_1R_1 = S_2R_2 = S_3R_3 = 0$ は満たされている.)この設計では Y_1, Y_2, Y_3 をハザードフリーに求め, 式の変形によりＳＲラッチの入力を求めたのと, ＳＲラッチによる設計法でＳＲラッチの入力を求めたのと, 結果は同じになる.

３．５．２ ダブルエッジトリガ型ＪＫフリップフロップ

ダブルエッジトリガ型フリップフロップは, クロックCの立上がりと立下がりの両方で入力に応答するフリップフロップである[5].

表３．８　エッジセンシングマスタスレーブ型ＪＫフリップフロップ

（a）遷移表

現在の状態	CJK 000	001	011	010	100	101	111	110	出力 z_1z_2
a	ⓐ	ⓐ	ⓐ	ⓐ	e	e	b	b	00
e	a	a	a	a	ⓔ	ⓔ	ⓔ	ⓔ	00
b	c	c	c	c	ⓑ	ⓑ	ⓑ	ⓑ	01
c	ⓒ	ⓒ	ⓒ	ⓒ	f	d	d	f	11
f	c	c	c	c	ⓕ	ⓕ	ⓕ	ⓕ	11
d	a	a	a	a	ⓓ	ⓓ	ⓓ	ⓓ	10
	次の状態								

（b）励起表

	$y_3y_2y_1$	CJK 000	001	011	010	100	101	111	110
a	000	000	000	000	000	100	100	001	001
b	001	011	011	011	011	001	001	001	001
c	011	011	011	011	011	111	010	010	111
d	010	000	000	000	000	010	010	010	010
e	100	000	000	000	000	100	100	100	100
	101	ddd	ddd	ddd	ddd	ddd	ddd	ddd	ddd
f	111	011	011	011	011	111	111	111	111
	110	ddd	ddd	ddd	ddd	ddd	ddd	ddd	ddd
		$Y_3Y_2Y_1$							

表３．８　エッジセンシングマスタスレーブ型
ＪＫフリップフロップ（続き）

（c）S_1R_1の制御表

$y_3y_2y_1$	CJK 000	001	011	010	100	101	111	110
000	0d	0d	0d	0d	0d	0d	10	10
001	d0	d0	d0	d0	d0	d0	d0	d0
011	d0	d0	d0	d0	d0	01	01	d0
010	0d	0d	0d	0d	0d	0d	0d	0d
100	0d	0d	0d	0d	0d	0d	0d	0d
101	dd	dd	dd	dd	dd	dd	dd	dd
111	d0	d0	d0	d0	d0	d0	d0	d0
110	dd	dd	dd	dd	dd	dd	dd	dd

S_1R_1

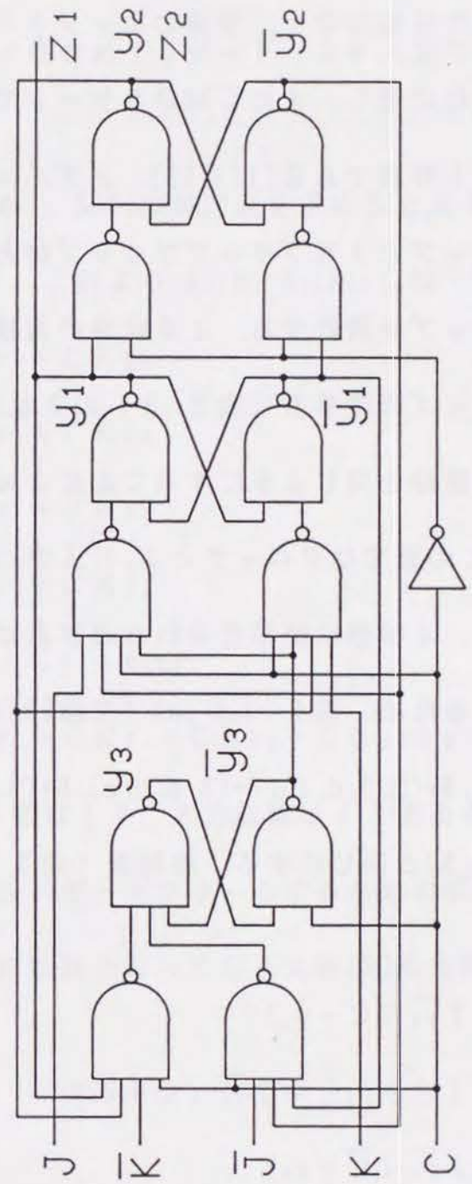


図３．１１　エッジセンシングマスタスレーブ型
ＪＫフリップフロップ

この型のフリップフロップはクロックの両方のエッジを使用するのでクロックの無駄な反転がなく，普通のエッジトリガ型フリップフロップに比べ高速化に適し，またＣＭＯＳゲートで実現するときには消費電力の点でも有利である[11][12]．ダブルエッジトリガ型にもＤフリップフロップとＪＫフリップフロップがあるが，ここではＪＫフリップフロップを設計する．１６状態の遷移表はエッジトリガ型フリップフロップの遷移表（表３．３）のクロックの立ち下がり部分を立ち上がり部分と同じようにすることによって容易に導かれる（表３．９）．この表ではクロックとＪ，Ｋ入力は同時に変化しないと仮定している．４状態に簡単化された遷移表は２種類あるが，設計結果を比較するため，表３．１０(a)（文献[5]の表８Ｂ）を用いる．(A=(0,1,4,5),B=(2,3,d,f),C=(8,a,c,e),D=(6,7,9,b)である．)状態割当ても文献[5]と同じにする．励起表（表３．１０(b)）より，Y_1, Y_2, Qは

$$
\begin{aligned}
Y_1 &= \overline{C}\,y_1 + J\,\overline{y_2}\,(C + y_1) \\
&\quad + \overline{K}\,y_2\,[C + y_1] + J\,\overline{K}\,(C + y_1) \\
Y_2 &= C\,y_2 + J\,\overline{y_1}\,(\overline{C} + y_2) \\
&\quad + \overline{K}\,y_1\,[\overline{C} + y_2] + J\,\overline{K}\,y_2 \\
Q &= [\overline{C}\,y_1] + [C\,y_2] \\
&\quad + [\overline{K}\,y_2\,(C + y_1)]
\end{aligned}
\qquad (3.25)
$$

となる[5]．(3.25)式をＮＡＮＤ３段で構成した回路図から $\overline{y_1}, \overline{y_2}$のためのインバータを含め１５ゲート，３９入力であり，遅れは４ゲート分である．

表３．１０(b)から S_1R_1の制御表を求めると表３．１０(c)となり（S_2, R_2も同様），これより $S_1, R_1; S_2, R_2, Q$は

$$
\begin{aligned}
S_1 &= C\,J\,\overline{y_2} + C\,\overline{K}\,y_2 \\
R_1 &= C\,\overline{J}\,\overline{y_2} + C\,K\,y_2 \\
S_2 &= \overline{C}\,J\,\overline{y_1} + \overline{C}\,\overline{K}\,y_1 \\
R_2 &= \overline{C}\,\overline{J}\,\overline{y_1} + \overline{C}\,K\,y_1 \\
Q &= \overline{C}\,y_1 + C\,y_2 + y_1 y_2 \\
&= C\,K\,y_1 + C\,K\,y_1 + C\,K\,y_2 + C\,K\,y_2 + y_1 y_2
\end{aligned}
\qquad (3.26)
$$

となり，ゲート数は１４，入力総数は４３である．この回路（図３．１２）は出力Ｑをハザードフリーにするための項 y_1y_2に必要なゲートを除いて，対称回路となっており文献[5]の回路より実用上優れている．

表３．９　ダブルエッジトリガ型ＪＫフリップフロップの遷移表

現在の状態	ＣＪＫ 000	001	011	010	110	111	101	100	出力Q
0	⓪	1	3	2				4	0
1	0	①	3	2			5		0
2	0	1	3	②	e-				0
3	0	1	③	4		f-			0
4	0				6	7	5	④	0
5		1			6	7	⑤	4	0
6				a-	⑥	7	5	4	0
7			b-		6	⑦	5	4	0
8	⑧	9	b	a				c	1
9	8	⑨	b	a			5-		1
a	8	9	b	ⓐ	e				1
b	8	9	ⓑ	a		7-			1
c	8				e	f	d	ⓒ	1
d		1-			e	f	ⓓ	c	1
e				a	ⓔ	f	d	c	1
f			3-		e	ⓕ	d	c	1

次の状態

（注）空欄は don't care 示す．また，状態の後の－はその状態の出力が dont' care であることを示す．

表３．１０　ダブルエッジトリガ型ＪＫフリップフロップ

（ａ）簡単化された遷移表

現在の状態	ＣＪＫ 000	001	011	010	110	111	101	100
(0145) A	Ⓐ,0	Ⓐ,0	B,0	B,0	D,0	D,0	Ⓐ,0	Ⓐ,0
(23df) B	A,0	A,0	Ⓑ,0	Ⓑ,0	C,1	Ⓑ,1	Ⓑ,1	C,1
(8ace) C	Ⓒ,1	D,1	D,1	Ⓒ,1	Ⓒ,1	B,1	B,1	Ⓒ,1
(679b) D	C,1	Ⓓ,1	Ⓓ,1	C,1	Ⓓ,0	Ⓓ,0	A,0	A,0

次の状態，出力

（ｂ）励起表

	y_1y_2	ＣＪＫ 000	001	011	010	110	111	101	100
A	00	000	000	010	010	100	100	000	000
B	01	000	000	010	010	111	011	011	111
C	11	111	101	101	111	111	011	011	111
D	10	111	101	101	111	100	100	000	000

Y_1Y_2Q

（ｃ）S_1R_1の制御表

y_1y_2	ＣＪＫ 000	001	011	010	110	111	101	100
00	0d	0d	0d	0d	10	10	0d	0d
01	0d	0d	0d	0d	10	0d	0d	10
11	d0	d0	d0	d0	d0	01	01	d0
10	d0	d0	d0	d0	d0	d0	01	01

S_1R_1

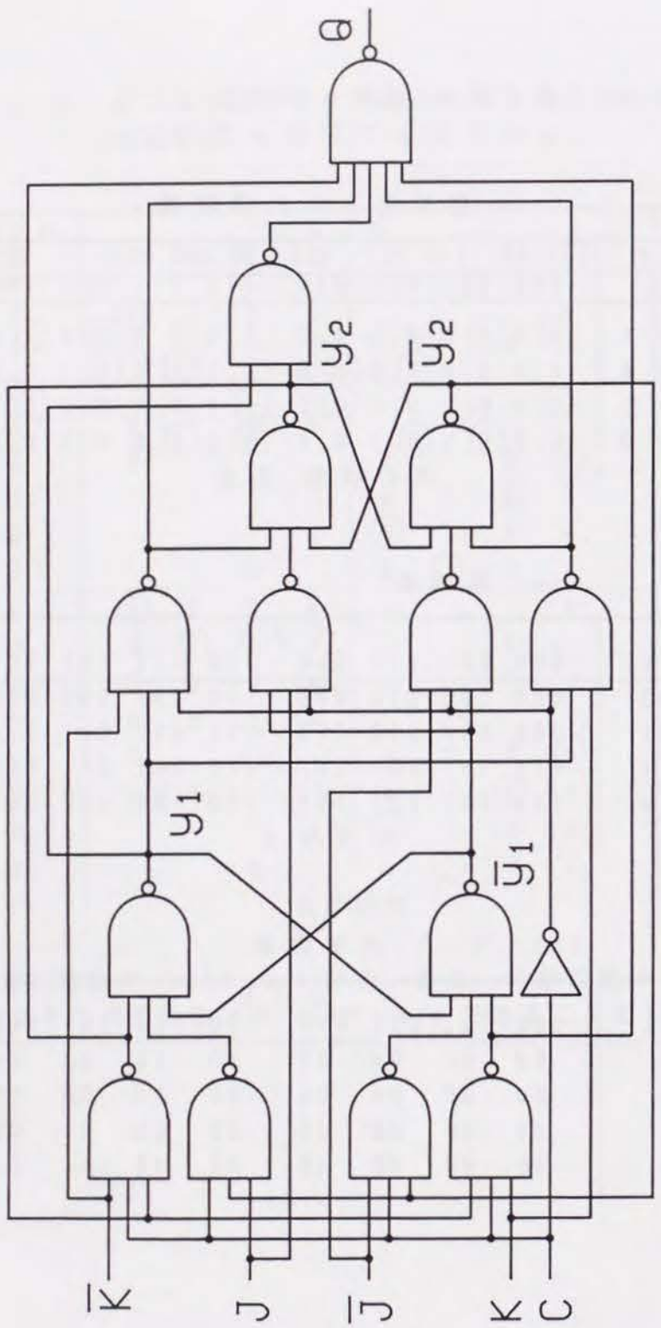


図３．１２　ダブルエッジトリガ型ＪＫフリップフロップ

３．６　おわりに

Ｄラッチ，Ｄフリップフロップ，ＪＫフリップフロップを設計し，経験的に設計されたと思われる回路も非同期理論に基づいて設計できることを示した．その過程で次の点を明らかにした．

まず，異なる回路のように見えるＥａｒｌｅラッチとＤラッチ，およびマスタスレーブ型とエッジトリガ型のＤフリップフロップがそれぞれ１つの遷移表から設計できる．したがって論理的に同じである（ゲートの遅れを考慮すると全く同じとは言えない）．

つぎに，ＪＫフリップフロップのエッジトリガ型とマスタスレーブ型の定義を明確にし，定義から広く使われている回路が理論的に設計できることを示した．クロックが１の間に入力を変化させてはならない型を導入し，この型の回路を３種類設計した．出力のタイミングを除いて，これら３つの回路は論理的に等価である．このうち２つはエッジトリガ型，他の１つはマスタスレーブ型と呼ばれているが，本来のエッジトリガ型，マスタスレーブ型とは異なることを明らかにした．同期回路で使用するときタイミングが入力変化の制限条件を満たしていれば，回路構成が簡単であるから，これらの回路を有効に利用できる．しかし，これらの回路は本来のエッジトリガ型，マスタスレーブ型とは論理的に動作が異なる．

最後に特殊なフリップフロップを，第２章で示したＳＲラッチに

よる設計法を適用して，２種類設計した．クロックの立上りで入力を取り込み立下りで出力が変化するエッジセンシングマスタスレーブ型ＪＫフリップフロップとクロックの立上りと立下りの両方で入力に応答するダブルエッジトリガ型ＪＫフリップフロップは前述のフリップフロップに比べかなり複雑な回路構成になる．

本章では広く使われているほとんどのフリップフロップが導かれ，その過程で各種の回路の同一性と相違点が明確にされた．これらの点を明確にしておくことは理論的にも実用的にも重要であると考えられる．

参　考　文　献

[1] 井上訓行，奥川峻史：非同期回路理論によるラッチとフリップフロップの設計，電子通信学会技術研究報告,CPSY86-58(1987).

[2] S.Okugawa and N.Inoue: Systematic Design of D Flip-flops Using Two State Variables, Electronics Letters Vol.23,No.17,pp.909-910(1987).

[3] F.J.Hill and G.R.Peterson: Introduction to Switching Theory and Logical Design, 3rd ed.,John Wiley & Sons, New York(1981).

[4] 井上訓行，奥川峻史：ＳＲラッチによる非同期回路の設計，

情報処理学会論文誌,Vol.28,No.10,pp1051-1059(1987).

[5] S.H.Unger: Double-Edge-Triggered Flip-flops,IEEE Transactions on Computers,Vol.C-30,No.6,pp.447-451(1981).

[6] S.R.Kunkel and J.E.Smith: Optimal Pipelining in Supercomputers,Proceedings of 13th International Symposium on Computer Architecture,pp.404-411(1986).

[7] G.A.Maley and J.Earle: The Logic Design of Transistor Digital Computers,Prentice-Hall,Englewood Cliffs(1963).

[8] E.Kjelkerud and O.Thessen: Methods of Modelling Digital Devices for Logic Simulation, Proceedings of 16th Design Automation Conference,pp.235-241 (1979).

[9] S.H.Unger: Asynchronous Sequential Switching Circuits, Wiley-Inter-Science,New York(1969).

[10] S.Muroga(室賀三郎,笹尾勤訳)：論理設計とスイッチング理論，共立出版(1981).

[11] Shin-Lien LU and Milos Ercegovac: A Novel CMOS Implementation of Double-Edge-Triggered Flip-Flops,IEEE Journal of Solid-State Circuits,Vol.25,No.4,pp1008-1010 (1990).

[12] M.Afghahi and J.Yuan:Double-Edge-Triggered D-Flip-Flops

for High-Speed CMOS Circuits, IEEE Journal of Solid-State Circuits, Vol. 26, No. 8, pp1168-1170(1991).

[13] Texas Instruments: The Bipolar Digital Integrated Circuits Data Book(1976).

第４章　マイクロプロセッサの設計開発システム

４．１　はじめに

非同期回路は第２章，第３章で述べたようにハザードやレースの問題があるため設計が複雑になり，大規模な回路の開発は難しい．順序回路は記憶素子に第３章で設計されたフリップフロップを用い，それに共通のクロックパルスを供給することによって同期式順序回路として設計できる．本章では最も複雑な論理システムであるマイクロプロセッサを同期回路で設計するシステムについて述べる．

論理システムはその仕様が与えられたとき，①ハードウェア（論理回路），②ファームウェア（マイクロプログラム），③ソフトウェア（プログラム）によって実現される．これら３つが適切に役割を分担したシステムを構成しなければならない[1]．また，システムは固定した状態を保持する必要はなく，可変構造プロセッサやダイナミックマイクロプログラミングのように，必要に応じて構成要素が変化するように構成できる．このような役割分担を評価するため，プログラム可能なＬＳＩを用いたマイクロプロセッサ開発システムが構築されている．

半導体集積回路技術の進展により大規模のＦＰＧＡ（Field Programmable Gate Array）が利用可能になり，これらを利用すれば

大学等の研究室内ににおいてもマイクロプロセッサなどの必要な機能を持つＬＳＩが開発できるようになってきている．ＦＰＧＡを用いたシステムは，市販のプロセッサに比べると規模と性能の点で制限を受けるが，特定用途向きであれば必要な性能を持つプロセッサが開発されている[2]．

ここで述べるシステムはプロセッサチップにＦＰＧＡを用いて，研究室内で独自のマイクロプロセッサを開発するシステムであり，研究と教育に利用されている[3][4][5]．本章ではシステムの構成と，計算機設計とアーキテクチャ教育への応用について述べる．

4．2 開発システムのアーキテクチャ

研究室内でマイクロプロセッサを開発するシステムを実現する上で重要な2つの点について検討する．

4．2．1 プロセッサチップの選択

研究教育用のシステムでは，設計変更や設計途中での動作確認などに柔軟に対応するため，利用するＦＰＧＡは

① 設計を実験室内で直ちに書き込めること

② 何回でも書き換えられること

が必要である．

現在マイクロプロセッサを1チップ化できる程度のＦＰＧＡは数

種類入手可能である[6]．これらの中で上記の①②を満たすので，Ｘｉｌｉｎｘ社のＬＣＡ（Logic Cell Array）が最も適切であると考えられる．ＬＣＡは組合せ回路とフリップフロップからなる論理ブロックがアレイ状に配置され，その間に配線領域がとられている[14]．論理ブロックと配線領域はＲＡＭ構造になっているので，構成(Configuration)データを外部からダウンロードするだけで直ちに設計をプログラムでき，書き換え回数に制限はない．

4．2．2 制御方式の検討

計算機の制御方式はマイクロプログラム制御方式と結線制御(Wired Logic Control)方式に大別されるが，設計とデバッグの点から，結線制御方式に比べ，マイクロプログラム制御方式には次の特徴がある[7]．

① 命令セットの設計に自由度が高く，単純なハードウェアでもいろいろな命令セットを実現できる．特にスッタク機構や割込み制御も複雑な制御回路を組み込まずに実現できる．

② 結線制御に比べ設計が容易である．特に設計変更に柔軟に対応できる．命令セットの一部を実装して評価し，必要に応じて命令を追加できる．

③ レジスタトランスファーレベルのデバッグが容易である．外部からマイクロ命令を与えることによってマイクロ命令で指定可能な

すべてのレジスタトランスファ操作を単独でテストできる.

④ マイクロアセンブラなどの汎用の開発支援ツールを用意できる.

これらの点はシステムを教育に用いるとき特に有効である. しかし, 性能と制御メモリの点で問題がある.

開発されたシステムはプロセッサチップにＬＣＡを用い, 制御メモリを外付けにしたマイクロプログラム制御方式を採用している. 結線制御方式のためのデバッグ環境はないがＬＣＡに制御部を入れることは可能である. また, 4.5でマイクロプログラム制御方式を結線制御方式に変換する方法を検討している.

4.3 ハードウェア構成

4.3.1 MDボードのハードウェア

マイクロプログラム制御方式の計算機を設計対象として開発した開発支援システムの構成について述べる[3][4]. 本システムは設計された計算機を実現するボード(ＭＤ(Microprocessor Development)ボードと呼ぶ), デバッグボード, ＭＤボードを支援するパーソナルコンピュータから構成されている.

ＭＤボードはプロセッサ用チップ（ＬＣＡ）と外付けの制御メモリ, 主メモリから成り, 図4.1に示すように接続されている. 制御メモリと主メモリはＲＡＭ(8x32Kbits)またはそれとピン互換のＲＯ

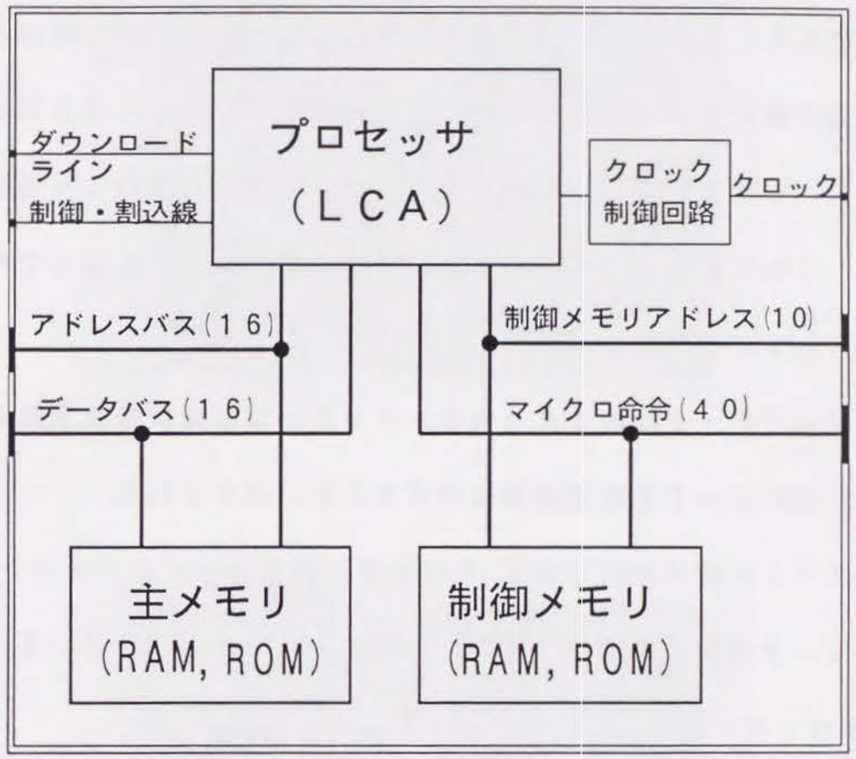


図４．１ ＭＤボード
(Microprocessor Development Board)

Mが使用されてる．構成要素を全てＲＡＭ構造にできる点が本ボードの大きな特徴である．このときＲＡＭにデータを書き込む手段が必要である．ＲＡＭ上でデバッグを行い，設計が完了した部分からＲＯＭに固定する方法を採っている．ＭＤボードには入出力部はなく，デバッグボードとパーソナルコンピュータの全面的な支援を受ける．このためにボード上の接続線はすべてコネクタを通して外部に出されている（図４．２）．

デバッグボードは約１００個のスイッチとほぼ同数の表示燈から成り，ＭＤボードを直接制御観測するために使用される．

システム全体の接続を図４．３に示す．ＭＤボードとパーソナルコンピュータの間は図に示すように３種類の線で接続されている．

① ダウンロード線

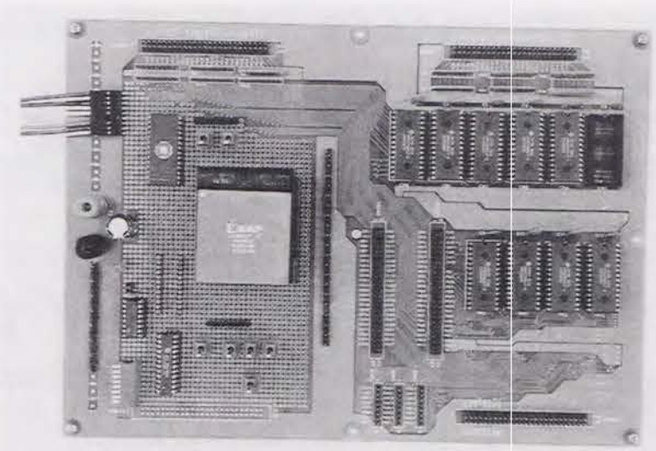
ＬＣＡのチップ内に外部からの構成データを受け取る機能があり，この線を用いてＬＣＡの構成(Configuration)を行う．

② 計算機の入出力線

ＭＤボードのデータバスが直並列変換器を通してパーソナルコンピュータの直列入出力ポートに接続され，パーソナルコンピュータがＭＤボード上に開発されたプロセッサの入出力装置となる．

③ デバッグ線

ＭＤボードから出ているすべての線がパーソナルコンピュータの



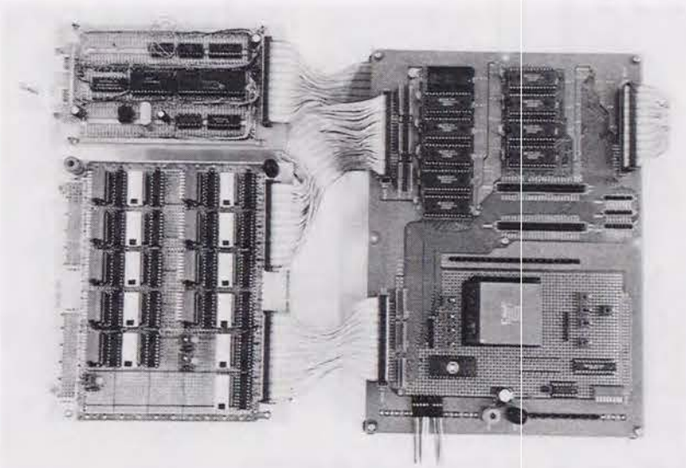


図４．２　ＭＤボードの写真

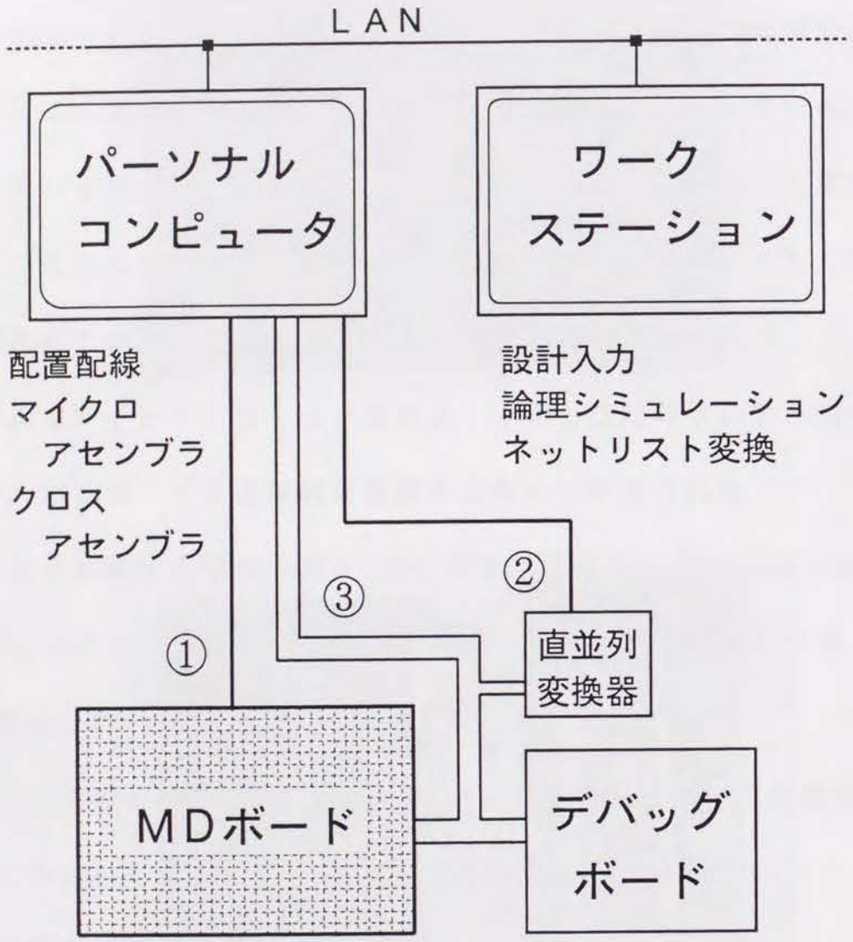


図４．３　システムの構成図

並列入出力ポートに接続され，デバッグ時に必要なデータの入出力に使用される．マイクロ命令やクロックもここから供給できる．

４．３．２　ＭＤボードの動作

ＭＤボードには次の５つの動作モードがある．

① Ｍ０：コンフィグモード

電源投入時にＬＣＡをコンフィギュアするモードで，スイッチによりパーソナルコンピュータまたはＲＯＭのデータが使用される．

② Ｍ１：マイクロ命令ロードモード

ＬＣＡ内のロジックによりパーソナルコンピュータからマイクロプログラムを制御メモリへロードする．

③ Ｍ２：機械語ロードモード

マイクロプログラムによって機械語を主メモリにロードする．

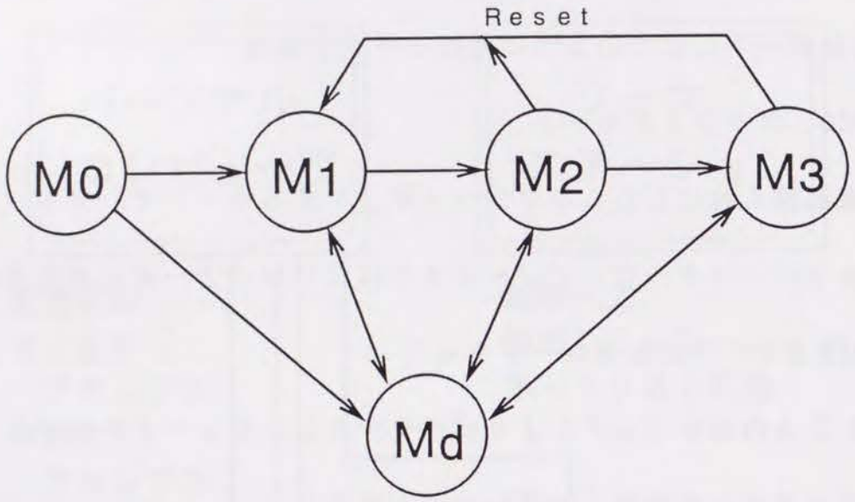
④ Ｍ３：ノーマルモード

機械語が実行される通常の動作をする．

⑤ Ｍｄ：デバッグモード

制御メモリの代わりにデバッグボード上のスイッチからマイクロ命令が供給される．このモードによりマイクロ命令で指定可能なすべてのレジスタトランスファ操作のデバッグが可能になっている．

モード間の遷移は図４．４のようになる．Ｍｄへはどのモードから



Ｍ０：コンフィグモード
Ｍ１：マイクロ命令ロードモード
Ｍ２：機械語ロードモード
Ｍ３：ノーマルモード
Ｍｄ：デバッグモード

図４．４　ＭＤボードの動作モード

でも移行できまたどのモードへ戻ることもできる．②と③はＲＡＭにデータを書き込むために使用され，必要がないときはパスする．

４．３．３　マイクロ命令と機械語のロードモード

マイクロ命令ロードモードの状態図を図４．５に示す．この時点ではプロセッサの必要なマイクロ操作はデバッグ済みで正常に動作すると仮定する．制御メモリは８ビットのチップが最大５個（４０ビット）使用できるので，このアドレス指定には制御メモリアドレスレジスタ（ＣＭＡＲ）を用い，チップを指定するためチップカウンタＣＣを用意する．パーソナルコンピュータから読み込まれたデータ（８ビット）をＬＡＴに入れ，このデータを順にＣＭＡＲの指定するアドレス(ｉ)のチップカウンタの指定するチップ(ｊ)にストアする．終了はＣＭＡＲおよびＣＣの指定した値を利用する．このモードの制御は結線制御であり，プロセッサと同時に設計するが標準的な回路も用意されている．

機械語ロードモードの動作はＩＰＬ(Initial Program Loader)をマイクロプログラム化したものである．バイトアドレスの場合のマイクロプログラム例を図４．９に示す．先頭でＳＷが検査され，１なら命令取出しサイクル(Fetch Cycle)に移り通常の命令が実行される．０ならパーソナルコンピュータから送られるデータ（命令）がプログラムカウンタ（ＰＣ）の指定するアドレスへ順に入れられる．転

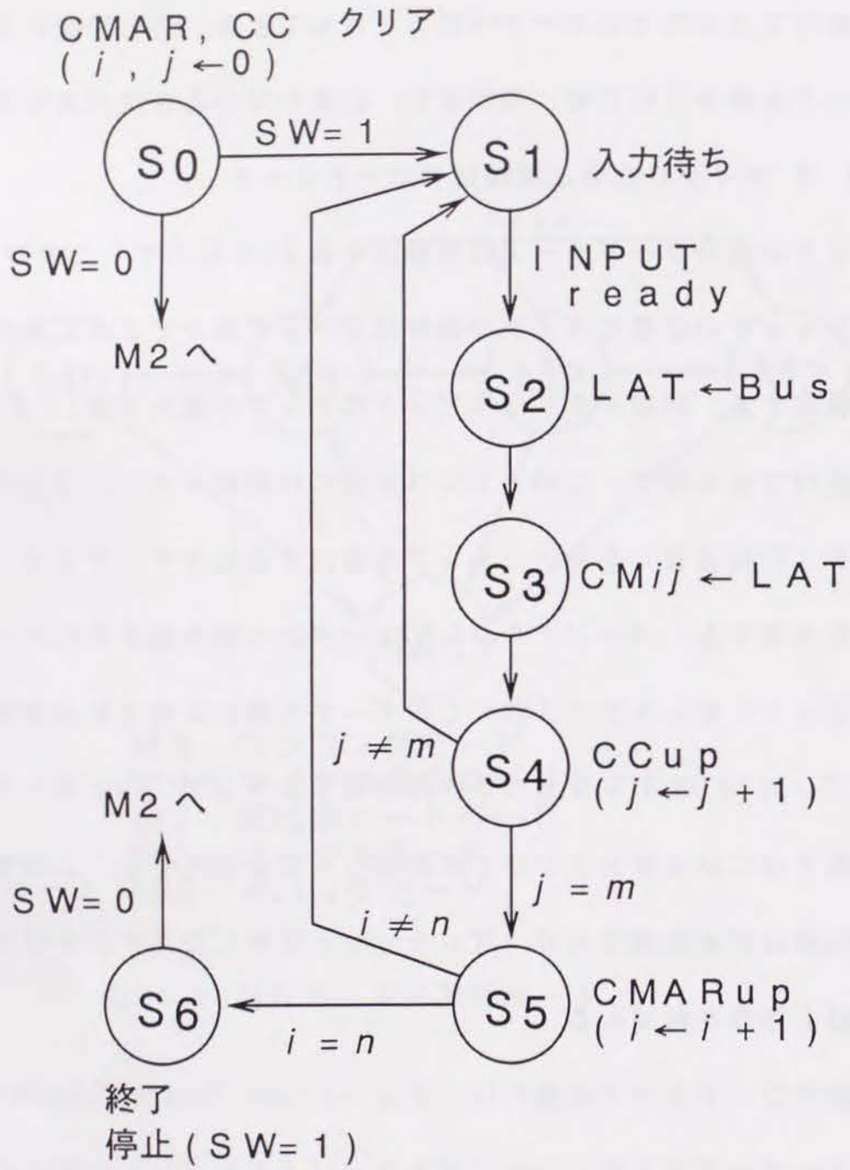


図4．5 マイクロ命令ロードモードの状態遷移図

送が終了すると入力待ちの状態になるが，ＳＷを検査してノーマルモードへ移ることもできる．

このように各モードで次のモードに移る準備をするのは，パーソナルコンピュータやワークステーションが電源投入後立上がるまでのプロセスと同様である（図4．6）．

4．4 ソフトウェア構成

ソフトウェアはＣＡＤシステムと開発デバッグ支援システムの2つである．ＣＡＤシステムは設計入力，論理シミュレーションおよび合成（ＬＣＡへのマッピング）の3つの部分から成っているが，いずれもパーソナルコンピュータおよびワークステーション上の市販のＣＡＤシステムを利用している．

開発デバッグ支援システムは ①マイクロアセンブラ ②クロスアセンブラ ③デバッグツールから構成されており，マイクロアセンブラがその中心になっている（図4．7）．

4．4．1 マイクロアセンブラ

本システムで使用するマイクロアセンブラは，設計によってターゲット計算機が異なるため機械独立型でなければならない．この型のマイクロアセンブラにはいくつかの種類があるが，デバッグのためにマイクロ命令と直接の対応が必要なことと，逆アセンブラを同

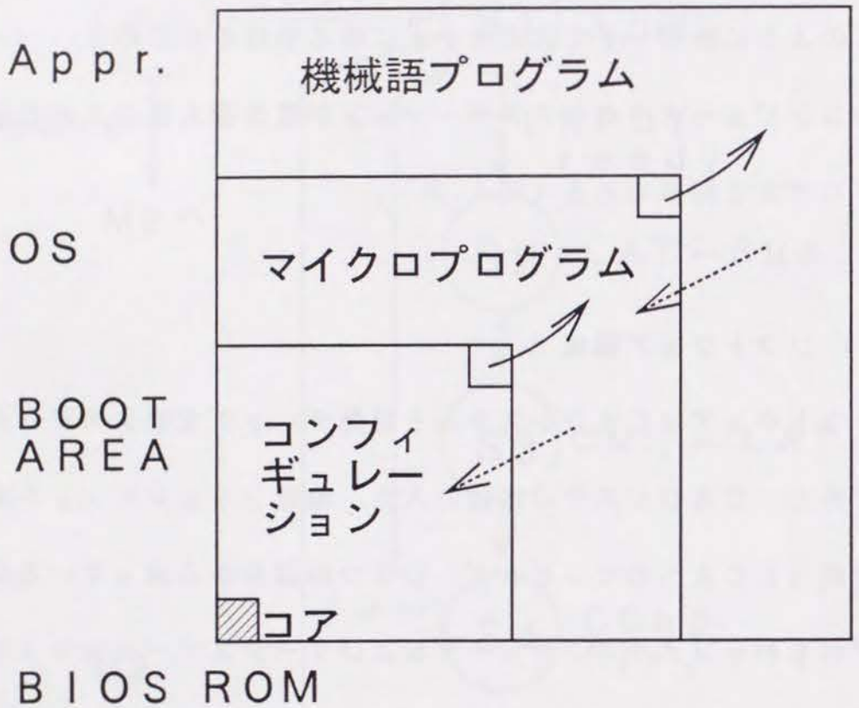


図４．６ ハードウェアの立上りシーケンス
（左側はソフトウェアの立上りシーケンス）

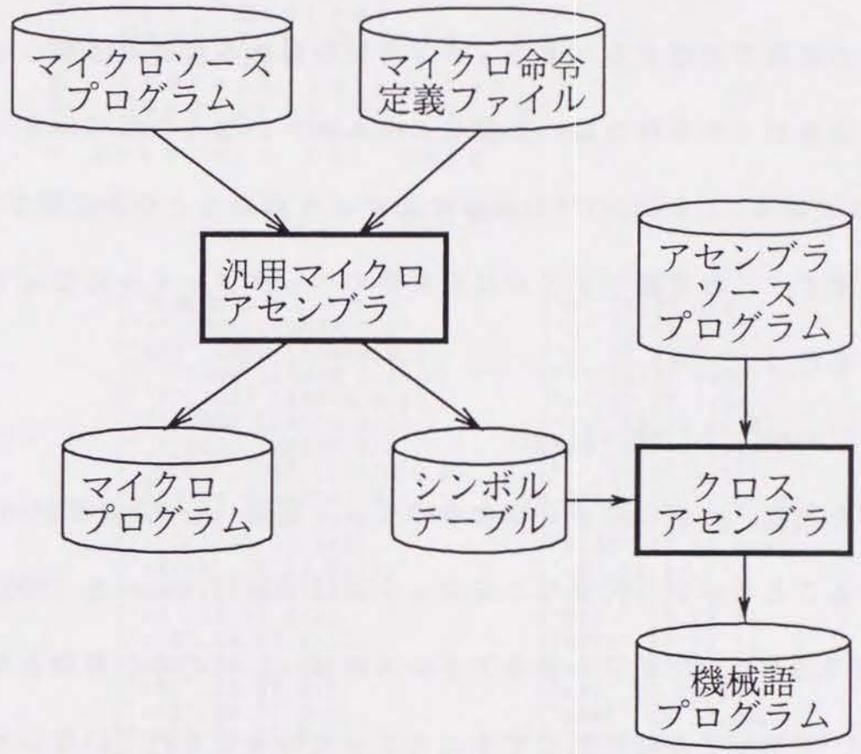


図４．７ ソフトウェア構成

時に開発するため，リスト形式[8]を用いる．リスト形式ではマイクロ命令の各フィールドに対しニモニックとビットパターンの対応表をマイクロ命令定義ファイルで与え，マイクロプログラムはニモニックの羅列で記述する．ドキュメント性を高めるため対応するマイクロ命令ビットを持たない冗長なニモニックが導入されている．図4．8に図4．16(a)のプロセッサのマイクロ命令とその定義ファイルを示す．この定義ファイルはC言語のヘッダファイルになっており，各行は

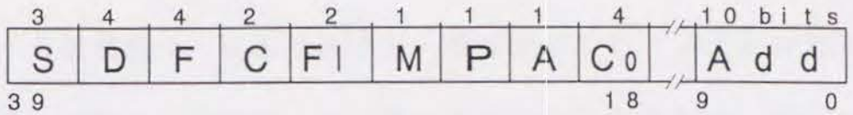
{"mnem", b1,b2,value}

の形をとり，mnemはマイクロ命令のビット位置 b1～b2の値がvalueであることを示す．冗長なニモニックでは b1,b2,valueを CODE_LENGTH,0,0とし，マイクロ命令アドレス部は b1,b2のみが有効となる．また，アセンブル時に対応するニモニックの指定されていないビット位置には0が入れられる．

次に例として図4．16(a)のプロセッサの機械語プログラムローダモードのマイクロプログラムを示す（図4．9）．四角で囲まれた部分が冗長なニモニックであり，これによってレジスタトランスファ形式に近い記述になっている．

4．4．2 クロスアセンブラ

クロスアセンブラはマイクロアセンブラが出力するシンボルテー



S Source Register
D Destination Register
F ALU&Shifter Function
C Carry Control
FI Flag Control
M Memory I/O Selection
P PC Count
A Address Selection
Co Condition Selection
Add Control Memory Address
8 bits are not used

(a) マイクロ命令

```
/* 冗長なニモニック */
 {"=",CODE_LENGTH,0,0},
 {"LAT",CODE_LENGTH,0,0},
 {",",CODE_LENGTH,0,0},
 {"IF",CODE_LENGTH,0,0},
 {"(",CODE_LENGTH,0,0},
 {")",CODE_LENGTH,0,0},
 {"_GOTO",CODE_LENGTH,0,0},
/* Sourse Register */
 {"sPCH",39,37,0x1},
 {"sPCL",39,37,0x2},
 {"sMARH",39,37,0x1},
 {"sMARL",39,37,0x2},
 {"sM",39,37,0x3},
 {"sIO",39,37,0x3},
 {"sR0",39,37,0x4},
 {"sR1",39,37,0x5},
 {"sR2",39,37,0x6},
 {"sR3",39,37,0x7},
/* Destination Register */
 {"NULL",36,33,0x0},
 {"dPCH",36,33,0x1},
 {"dPCL",36,33,0x2},
 {"dMARH",36,33,0x3},
 {"dMARL",36,33,0x4},
 {"dM",36,33,0x6},
 {"dIO",36,33,0x6},
 {"dLAT",36,33,0x7},
 {"dR0",36,33,0xC},
 {"dR1",36,33,0xD},
 {"dR2",36,33,0xE},
 {"dR3",36,33,0xF},
/* ALU & Shifter Function */
 {"SHR",32,29,0x4},
 {"RCR", 32,29, 0x5},
 {"SHL",32,29,0x6},
 {"RCL",32,29,0x7},
 {"",32,29,0x8},
 {"@",32,29,0x9},
 {"&",32,29,0xA},
 {"|",32,29,0xB},
 {"<",32,27,0x30},
 {"+",32,27,0x38},
 {"--",32,27,0x3C},
 {"++",32,27,0x33},
 {"-",32,27,0x37},
/* Other Function */
 {"C2_LD",26,25,0x1},
 {"FLAG",26,25,0x2},
 {"sM",24,24,1},
 {"dM",24,24,1},
 {"PC++",23,23,1},
 {"sPCH",22,22,1},
 {"sPCL",22,22,1},
 {"aPC",22,22,1},
 {"aMAR",22,22,0},
/* Condition Selection */
 {"C1",21,18,0x1},
 {"C2",21,18,0x2},
 {"Z",21,18,0x3},
 {"S",21,18,0x4},
 {"I",21,18,0x5},
 {"O",21,18,0x6},
 {"GOTO",21,18,0x7},
 {"INT9",21,18,0x9},
 {"BUS0",21,18,0xA},
 {"BUS1",21,18,0xB},
 {"INT12",21,18,0xC},
 {"INT13",21,18,0xD},
 {"SW",21,18,0xE},
 {"INST",21,18,0xF};

/* Bit 17-10 Not Used */
/* Label Field */
 Micro labelField ={
          NULL, 9,0, 0};
```

(b) (a)の定義ファイル

図4．8 マイクロ命令とその定義ファイル

```
:MAIN_LOOP
  IF (SW) _GOTO FETCH          ;SW:Mode Switch
:WAIT_IN_CHAR
  IF (I) _GOTO READ_CHAR       ;I:Input Flag
  GOTO WAIT_IN_CHAR
:READ_CHAR
  dR1 <= sIO
:WRITE_TO_MEMORY
  dM <= sR1, aPC               ;Addressed by PC
  PC++, GOTO WAIT_IN_CHAR
```

図４．９ 機械語ロードモードのマイクロプログラム

ブルから機械語コードを生成する．機械語のニモニックとして，その命令の制御メモリ内での実行開始番地に付けられたシンボルを使い，クロスアセンブラは命令語から制御メモリへのマッピング表を逆に用いて命令コードを決める．この方法は命令セットを定義せず，必要が生じた命令から定義して使うときに便利である．しかし，汎用性とソースプログラムのドキュメント性は必ずしも十分とはいえない．

４．４．３ デバッグ支援

ＭＤボードとパーソナルコンピュータの間のデバッグ線（マイクロ命令，アドレスバス，データバス）を使ってプロセッサチップを制御すると，任意のレジスタトランスファ操作を行うことができる．例えばレジスタＡにデータをセットするには，データバスに必要なデータを出して，マイクロ命令

A←databus

を実行する．この結果を見るには，マイクロ命令

null←A

を出して，データバスを観測する．ここで null は行き先レジスタなしを示す．

このようにして１ステップずつマイクロ命令を対話形式で実行し結果を観測すれば，初期のレジスタトランスファレベルのデバッグ

に非常に有効である．レジスタの出力はゲートのみを通してバスに出るので，観測には通常クロックを必要としない．したがって，どの時点で内部状態を観測しても次の動作には影響を与えず，観測バス[11]を設けたのと同じ機能を持つことになる．

マイクロプログラム制御方式のプロセッサでは，動作中のデータバスとマイクロ命令を同時に観測すると，バス上のデータの出所と行き先がわかる．バス上のデータをパーソナルコンピュータに取り込み，行き先レジスタの表示を更新すると，常にその時点の全てのレジスタの内容を表示しておくことができる．また，マイクロ命令はビット数が多い上にいろいろなフィールドに分かれているため，１６進や２進形式で入力せず，前述のマイクロアセンブラを用いてレジスタトランスファ形式の記述から２進形式のマイクロ命令を生成している．観測表示のときは逆アセンブラがマイクロ命令定義ファイルを参照してマイクロ命令をニモニック表示する．このデバッガにはつぎの４つのデバッグモードがある．

① シングルステップモード

マイクロ命令を１つずつ対話型で２進形式に変換して送出し，結果を表示する．このモードは最も初期のレジスタトランスファレベルのデバッグに使用され，クロックもパーソナルコンピュータから供給される．図４．１０(a)にデバッグ中の画面のハードコピ

ーを示す．上部のレジスタ表示部はレジスタの内容のみを更新し，下部のマイクロ命令は１行毎に入力し直ちに実行される．

② 連続モード

マイクロアセンブラによってファイルに生成されているマイクロ命令を，クロックとともに連続してプロセッサに供給する．このモードは機械語の作成，デバッグ時に使われる．

③ 観測モード

制御メモリ内のマイクロ命令を実行させて動作を観測する．表示のために逆アセンブラが走るので，観測できる速さはソースプログラムの複雑さに依存する．観測中の画面は図４．１０(b)に示すように１ステップが１６進表示とその逆アセンブル結果の２行に表示される．

④ 高速モード

動作は観測モードと同じであるが，高速化のため観測結果を表示せず，ファイルに取り込む．

４．５ プロセッサの開発手順

図４．３のシステムを用いてＭＤボード上にプロセッサを開発する手順の概略を図４．１１に示す．

① プロセッサの仕様（アーキテクチャ，命令の形式など）を決める．

```
      <<送信モード1>> [シングルステップ]

IR=0000 LAT=0881 ACC=2D7F  PC=0002 MAR=0000   M=0000 I/O=0000
R0=5AFF  R1=087F  R2=5AFF  R3=0881  R4=0000 FLG=0000 BUS=0881

ACC = R0
ACC = SHR ACC
LAT = ACC
LAT = LAT & R0
R1 = LAT
PC++ , NULL = PC
PC = R2
DSW
データバスへの出力値 [16進数] => [現在 5AFF] => 0000
PC = DSW
PC++ , NULL = PC
PC ++
NULL = PC
LAT = LAT + PC
R3 = LAT
```

(a) シングルステップモード

```
      <<受信モード1>> [逆アセンブル]

IR=0010 LAT=2A55 ACC=2A55  PC=1002 MAR=1002   M=0010 I/O=0000
R0=FF00  R1=0000  R2=0000  R3=0000  R4=0000 FLG=1000 BUS=1002

<001>    <07 00 07 00 00>
 FETCH    MAR = PC
<002>    <90 40 10 00 00>
          IR = M, PC++
<003>    <00 00 00 F0 00>
          GOTO INST
<010>    <41 10 01 F0 2A>
ADDR0     LAT = R0, GOTO ADDA
<02A>    <48 1A 00 FD 33>
ADDA      LAT = LAT+ACC, FLAG, GOTO STA
<133>    <29 10 00 F0 01>
STA       ACC = LAT, GOTO FETCH
<001>    <07 00 07 00 00>
FETCH     MAR=PC
```

(b) 観測モード

図４．１０ デバッグ中の画面

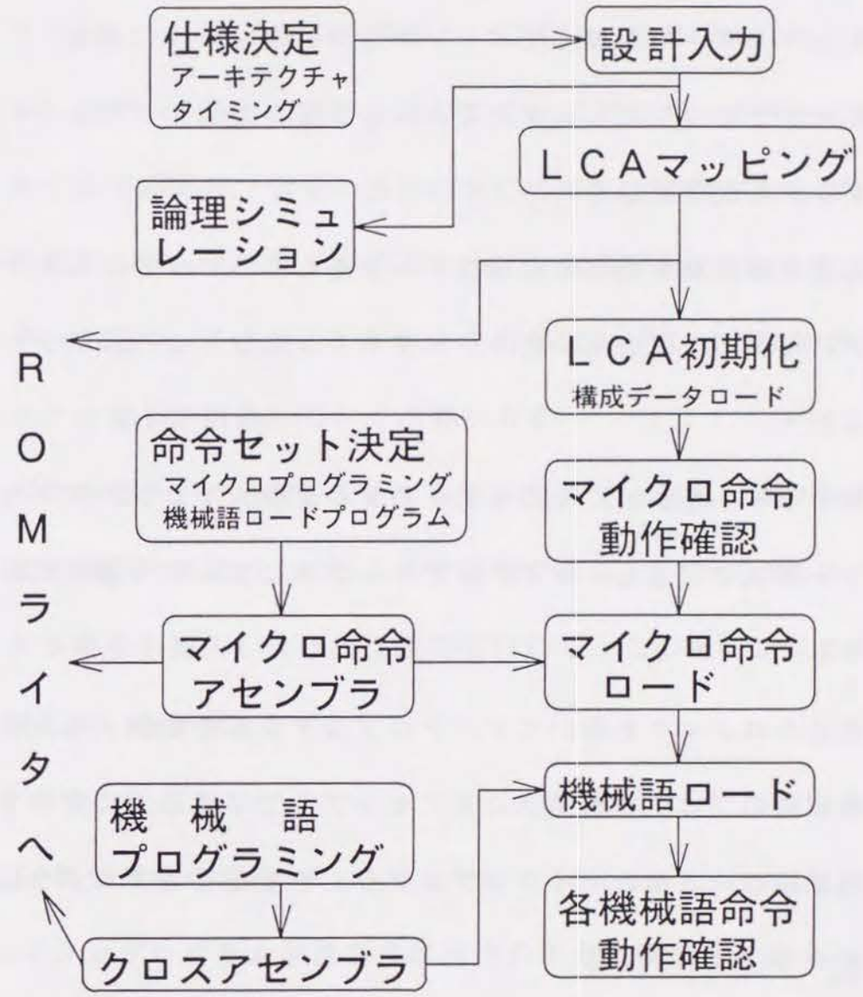


図４．１１ プロセッサの開発手順

② ワークステーション上で回路図入力し，論理シミュレーションを行う．

③ ネットリストを変換してパーソナルコンピュータに送る．

④ パーソナルコンピュータ上でロジックをＬＣＡのブロックにマッピングし配置配線を行う．

⑤ 配置配線の結果得られた構成データをＬＣＡにダウンロードする．

⑥ デバッグボードを用いてレジスタトランスファレベルのデバッグを行う．

⑦ 命令セットを決め，そのマイクロプログラムをレジスタトランスファ形式で記述し，マイクロアセンブラによって２進形式に変換する．

⑧ 組込みロジックを用いてマイクロプログラムを制御メモリに移す．

⑨ 機械語のプログラムをクロスアセンブラで２進形式に変換する．

⑩ 機械語ロード用のマイクロプログラムを起動してプログラムを主メモリにロードする．

②においてマイクロプログラムを制御メモリにロードするロジックを，⑦で機械語プログラムを主メモリにロードするマイクロプログラムを組み込んでおく必要がある（図４．６）．

プロセッサ設計上の制約

上述の方法でデバッグするためには，ＭＤボード上にプロセッサ

を設計する際に次の制限がある．

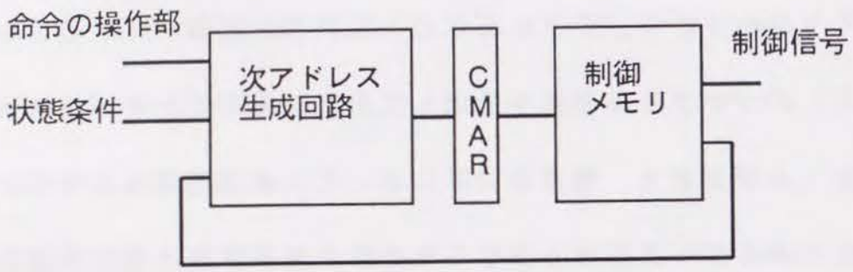
① チップの内部を観測するために，外部からデータを読むとき（メモリリード，Ｉ／Ｏリードなど）以外は，内部バス（のうちの１本）のデータを外部端子を通してチップ外のデータバスに出しておく必要がある．通常の設計においてこれは問題にならない．

② 外部からデータをセットするために，外部データをソースとして，データをセットされるレジスタを行き先レジスタに指定できるマイクロ命令を組込んでおく必要がある．

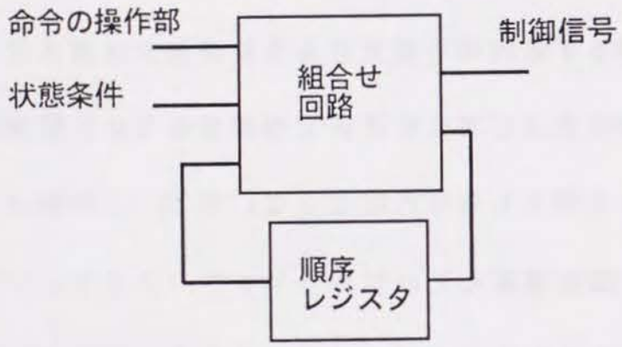
③ バスを通さずに内容を変更できるレジスタ（例えばカウンタ）の変更後の内容はバスに出るまで観測できない．観測のためにマイクロ命令を挿入しなければならないので，この種のレジスタは使用しない方が望ましい．

４．６ 制御方式の変換

マイクロプログラム制御方式は前述のように柔軟性に富み，設計，デバッグが容易であるが，実行速度と制御メモリに問題がある．現状ではＦＰＧＡに制御メモリを内蔵することは難しく，ゲートアレイ化する場合でも制御メモリを内蔵するとゲートの使用効率が悪くなる．プロセッサの制御はマイクロプログラムでも結線制御でも可能であり，この２つは実現方法が異なるだけで本質的には同じ順序



（a）マイクロプログラム制御



（b）結線制御

図４．１２ 制御部のブロック図

回路である．

マイクロプログラム制御と結線制御のブロック図を図４．１２に示す．図４．１２(a)の制御メモリは制御信号と，次アドレスと，次アドレスを決定するための選択信号とを生成する組合せ回路である．図４．１２(b)の組合せ回路は制御信号(順序回路の出力）と順序回路の次の状態を決める．この図から２つの制御方式が等価であることがわかる．すなわち結線制御の組合せ回路をＲＯＭで実現すれば（容量的に実現不可能であろうが）マイクロプログラム制御となり，マイクロプログラム制御の制御メモリで表されるロジック（真理値表に相当する）をランダムロジックで実現すれば結線制御となる．

ここではマイクロプログラム制御方式で検証済みのプロセッサを，データプロセッサ部（レジスタ，ＡＬＵ，バス構造など）は変更しないで，結線制御方式に変換する方法を検討する．マイクロアセンブラではレジスタトランスファ形式で記述された制御論理をマイクロプログラムに変換したが，レジスタトランスファ記述を結線制御に変換することもできる．また，一度マイクロプログラム制御で実現されたものを結線制御に変換することも可能である（図４．１３）．ここでは後者を採用し，次の２つの変換方法を検討する．

(1) 組合せ回路のみの変換

制御メモリは組合せ回路を真理値表の形で実現しているので，こ

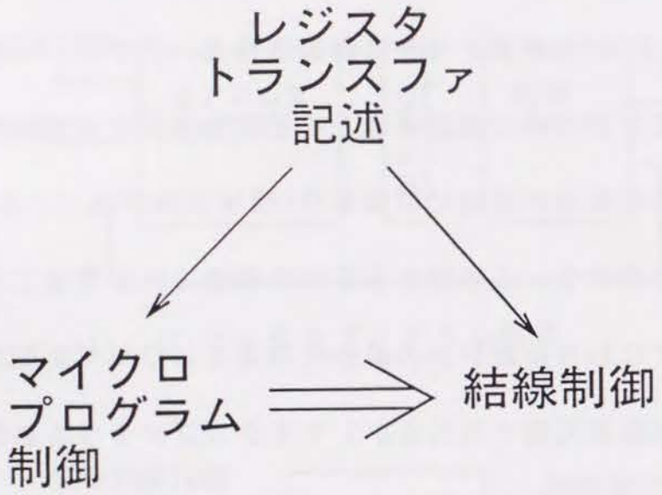


図４．１３ 制御の変換

の部分を最適化した組合せ回路で実現する．たとえば，図４．１４(a)のマイクロプログラムを２進形式に直すと図４．１４(b)になる．制御語には定義する必要のない部分が多数含まれている．制御メモリに書き込むときは０または１にするが，この図では don't care になっている．図４．１６(a)のデータプロセッサを図４．８のマイクロ命令で制御する場合では１０入力３２出力の組合せ回路を簡単化して実現することになるが，どの程度の回路になるか確認できていない．この場合，次アドレス生成回路とＣＭＡＲはそのまま使用することになる．

(2) 全体の変換

各々のマイクロ命令は出力と次の状態を決める情報を持っているので，直接状態遷移図（表）に書き換えることができる．図４．１５は図４．１４のマイクロプログラムを状態図に変換したものである．すべてのマイクロ命令をこのように書き換えると，マイクロプログラムの語数を状態数とする順序回路となる．しかし，これは出力が状態のみで決まるＭｏｏｒｅ形の順序回路である．一方，図４．１２(b)の回路は出力が状態と入力で決まるＭｅａｌｙ形の順序回路である．一般に前者の方が状態数が多くなる[9]ので，状態の最小化と論理最適化を行って順序回路として実現した場合，回路規模が問題である．

```
:MAIN_LOOP
   IF (SW) _GOTO FETCH          ;SW:Mode Switch
:WAIT_IN_CHAR
   IF (I) _GOTO READ_CHAR       ;I:Input Flag
   GOTO WAIT_IN_CHAR
:READ_CHAR
   dR1 <= sIO
:WRITE_TO_MEMORY
   dM <= sR1, aPC               ;Addressed by PC
   PC++, GOTO WAIT_IN_CHAR
```

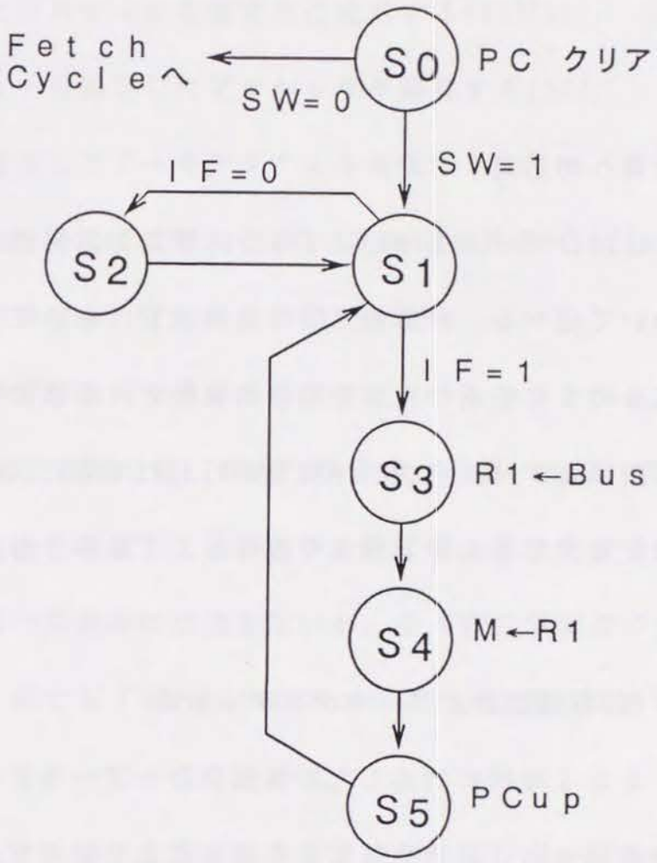
(a) レジスタトランスファ記述

```
ｱﾄﾞﾚｽ      マイクロ命令
0000   ---0000----------0-1110  xxxx
0001   ---0000----------0-0101  0011
0010   ---0000----------0-0111  0001
0011   0111101110000--00-0000  ----
0100   1010110110000--1010000  ----
0101   ---0000----------1-0111  0001
```

アドレスは4ビットに短縮されている.
xxxx はフェッチサイクルの開始番地を
- は don't care を表す.

(b) 2進表現

図4.14 マイクロ命令とその2進表現



PC　プログラムカウンタ
IF　入力フラッグ

図4.15 図4.14の状態遷移図による表現

現在(1)(2)の方法で実現可能な規模の回路が得られるか検討中である.

4.7 教育への応用

本節ではＭＤボードを中心とするシステムの計算機工学教育への応用について述べる. 計算機工学の実験実習において, 従来からゲートレベルのＩＣを用いた論理回路の実験と, 市販のマイクロプロセッサを用いたアーキテクチャ実習が行われてきた[10]. しかし, これだけでは大学等の計算機工学教育としては不十分になってきている.

4.7.1 計算機設計とアーキテクチャ教育

最近, ＬＳＩ時代に対応した計算機設計とアーキテクチャ教育の重要性が認識され, ＦＰＧＡなどの利用により研究室内でもプロセッサの開発が可能になったことを背景に, 計算機工学教育に対する議論が活発になり, また, 具体的な教育システムも開発されている[3]～[5][11]～[13]. これらを要約すると

(1) アーキテクチャ設計から実装までの統合教育

(2) アナリシス型からシンセシス型教育へ

(3) ＣＡＤシステムを用いたＬＳＩ設計教育

の必要性が強調されている.

また, 教育方法としては次の３つが考えられる.

(1) 開発したプロセッサを教育用に使用する[11][12].

(2) 命令セットを固定したプロセッサを開発する[12].

(3) 自由に設計したアーキテクチャを実現する[4].

各々の方法について文献[4][13]で詳細に検討され, (1),(2)について実施例が報告されている[11][12].

ここでは学習者の個性を尊重し, また, アーキテクチャ設計を重視して, (3)の方法をとる. 即ち, 学習者が目標, 知識, 興味, 能力, 時間, 労力などを考慮して自らアーキテクチャを決定する. この方法は多人数の一斉教育には適さないが, 少人数で学習者の個人差が大きい場合, 誰でも「それなりのプロセッサ」を完成できる点で非常に有効である.

設計されたアーキテクチャを本章で述べたＭＤボード上に実現する. 本システムを用いると上記の必要性(1)～(3)はすべて満たされており, 開発システムの設計環境とデバッグ環境が整っているので効果的である. (実習教育の詳細については文献[4]参照.)

4.7.2 開発されたプロセッサの例

つぎに実習教育において開発されたプロセッサの例を図４.１６に示す.

(a) は８ビット２バス構造のプロセッサである.

アドレスバスが１６ビットになっているためマイクロプログラムのステップ数が多くなる.

(b) は８ビット３バス構造のプロセッサで，汎用レジスタを１つにするとほぼ最小構成のプロセッサとなり，学生実習で設計するときの基本になると考えられる.このプロセッサを用いてKUE-CHIP[11]のエミュレーションが行われた.

(c) は１６ビット１バス構造の演算回路を高速化したプロセッサであり，πの値５０００桁を約３０分で計算できる(クロック４MHz).

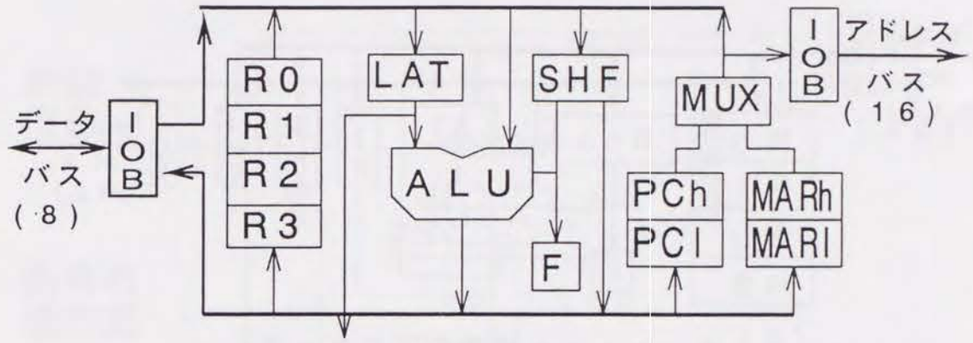
(d) は１６ビット２バス構造の単純なプロセッサであり,アドレス空間を１６ビットにするなら(a)より設計し易くマイクロプログラムも短くなる.

(e) は単純な形のマイクロプログラム制御部であり，(a)～(d)のプロセッサを制御できる．スタックを設けてマイクロサブルーチンを使用できる制御部や，アドレス部をデータとしてレジスタに取り込むことのできる制御部も開発されている.

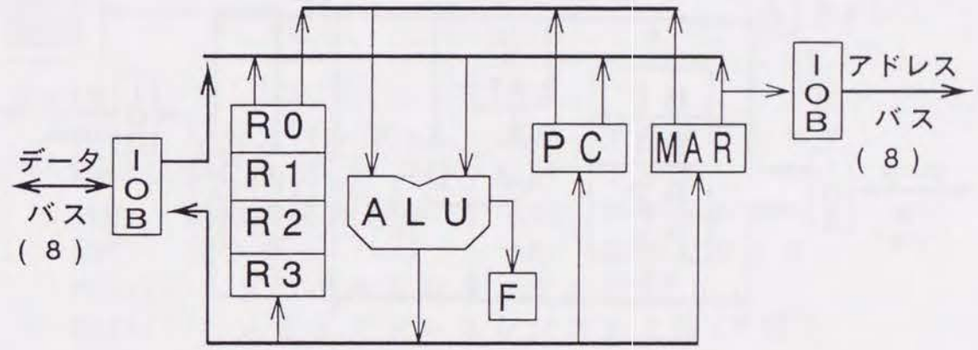
(f) は(d)のＬＣＡ(XC3090)内の配置配線図である.

４.７.３ 教育システムとしての評価

約４年間の実習教育に利用した結果，学習者が各自の条件にあったプロセッサを開発すれば，ほとんど完成させることができ，「独自の計算機」を完成させたときの満足感は大きく，将来の自信にも

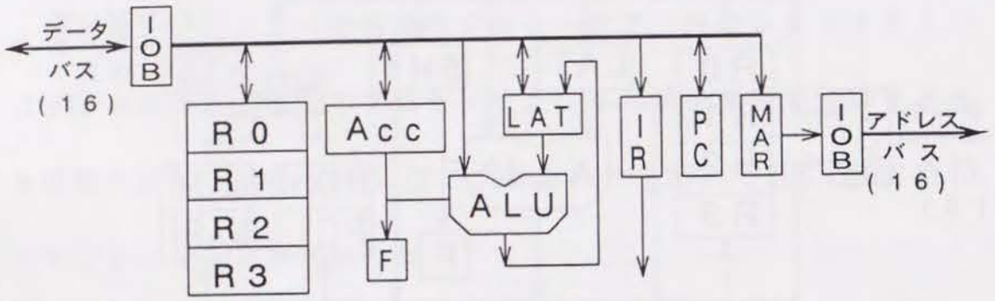


(a) ８ビット２バス構造のプロセッサ

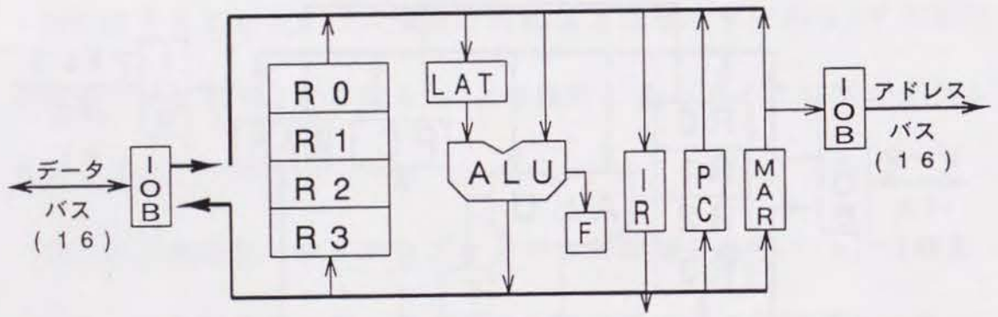


(b) ８ビット３バス構造のプロセッサ

図４.１６ 設計例

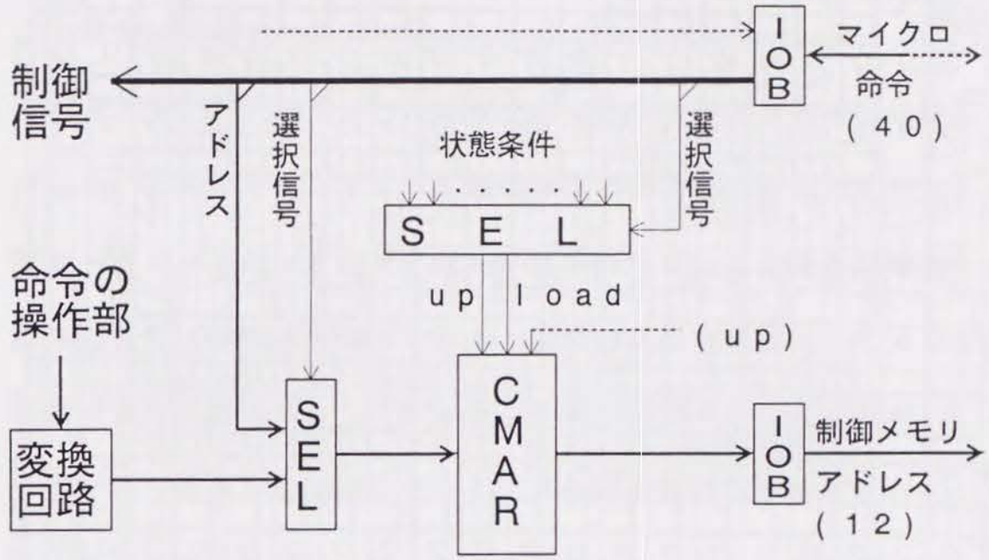


(c) 16ビット1バス構造のプロセッサ



(d) 16ビット2バス構造のプロセッサ

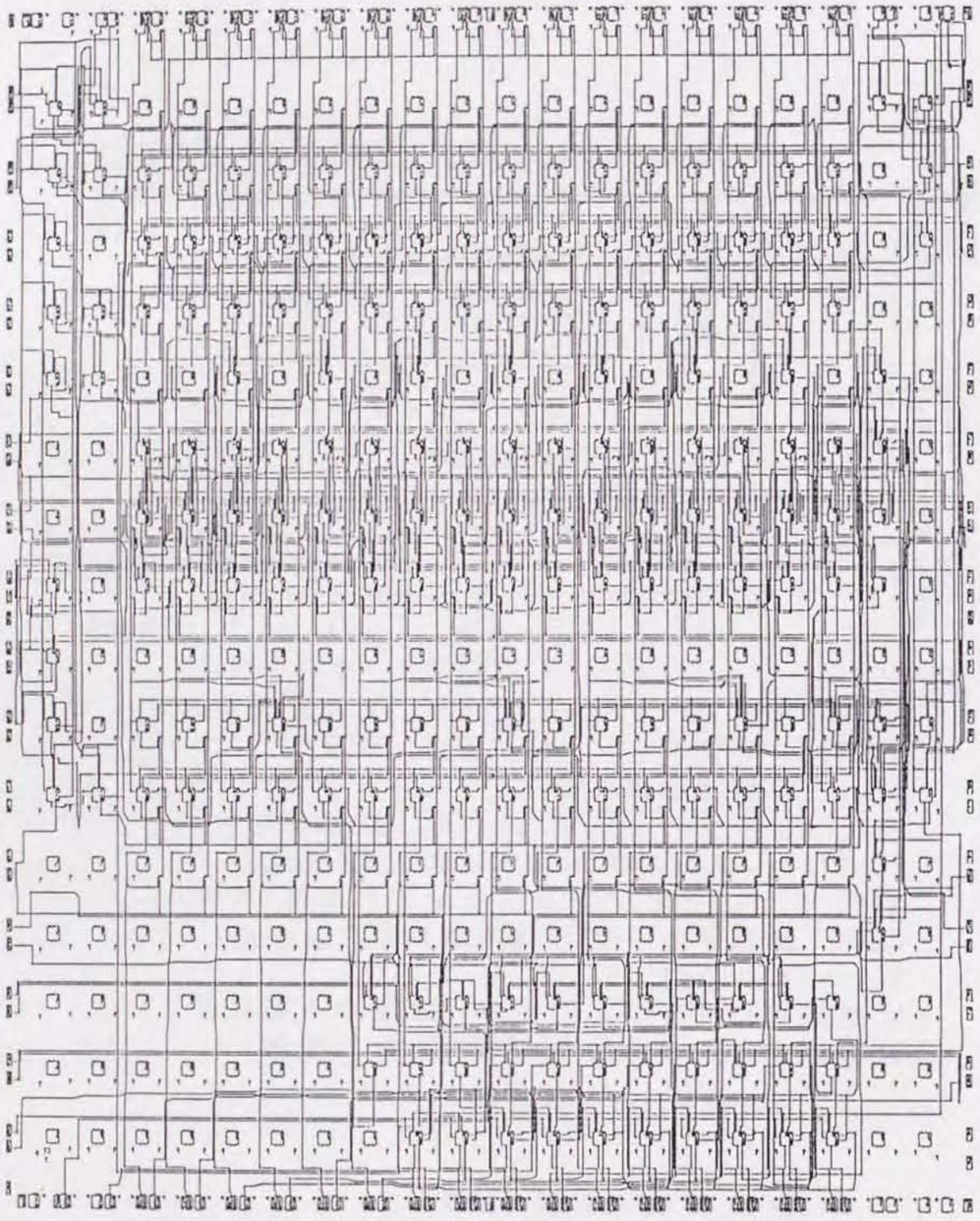
図4.16 設計例（続き）



IOB: 入出力バッファ　　MUX: マルチプレクサ
R0~R3: レジスタ　　LAT: ラッチ
ALU: 算術論理演算器　　Acc: アキュムレータ
SHF: シフタ　　IR: 命令レジスタ
PCh(l): プログラムカウンタ上位(下位)
MARh(l): メモリアドレスレジスタ上位(下位)
CMAR: 制御メモリアドレスレジスタ
SEL: セレクタ
状態条件: フラグ, 割込み, 命令の一部など
点線はマイクロ命令ロードのときに使用

(e) マイクロプログラム制御部

図4.16 設計例（続き）



(f) (d)のＬＣＡ(XC3090)内での配置配線図

図４．１６ 設計例（続き）

つながっている．また，プログラミングの段階になって機械語の重要性に対する認識が深まり，既存の計算機の命令セットを考え直すよい機会になっている．

教育システムとして用いた場合，本システムは制御メモリにＲＡＭを用いたマイクロプログラム制御方式の採用と，パーソナルコンピュータを利用したデバッグ環境が他の方法に比べて優れた特徴になっており，学部上級の半年から１年間の実習教育用として十分なシステムである．しかし，プロセッサチップにＦＰＧＡ（XC4010，１万ゲート相当）を使用していることから実装規模が制限され，ＬＳＩ化が論理回路部分に限られる問題がある．また，複雑な順序回路の設計実習のために結線制御方式のプロセッサの開発教育にも対応する必要がある．

４．８ おわりに

すべてＲＡＭ構造のＬＳＩ（ＬＣＡとＲＡＭ）から成るプロセッサ開発用のボードとその支援環境を開発し，計算機工学の実習教育に応用して効果を上げている．本システムを利用すると，設計したプロセッサを配線作業なしに直ちに実装し，実機で検証できる点が特に有効であり，多数のプロセッサが開発されている．

しかし，処理速度と利用できるゲート数の問題からまだ実用的な

プロセッサは開発されていない．ゲート数の問題を解決するにはチップの集積度の向上を待つと同時にＦＰＩＣ(Field Programmable Interconnect Device)[15]などを用いたマルチチップシステムに拡張する必要がある．また，処理速度の向上のためと，現状では制御メモリをチップに内蔵することは難しいために，結線制御化を考えなければならない．マイクロプログラム制御方式で検査済みのプロセッサの制御部を結線制御に変換する方法を検討した．これらの方法で変換した場合どの程度の規模の回路が得られるか確かめる必要がある．

参　考　文　献

[1] 赤星博輝，安浦寛人：計算機アーキテクチャ設計支援に関する考察，情報処理学会研究報告93-DA-68-7(1993).

[2] 冨田昌宏，澄川文徳，菅沼直昭，平野浩太郎：汎用エンジン ＲＭ－Ⅱの構成，情報処理学会論文誌,Vol.35,No.4,pp.636-644(1994).

[3] 井上訓行：ＦＰＧＡを用いたマイクロプロセッサ開発システムとその教育への応用，電子情報通信学会技術研究報告,CPSY94-58(1994).

[4] 井上訓行：計算機アーキテクチャ教育とその支援環境，情報処理学会論文誌，Vol.35,No.2,pp.155-163(1994).

[5] 古田勇次，岡田光博，井上訓行：計算機ハードウェア教育のための支援システムの開発，情報処理学会第４４回全国大会，2S-2(1992).

[6] Thomas R.Clark: Fitting Programmable logic,IEEE Spectrum, Vol.29,No.3,pp.36-39(1992).

[7] 萩原宏：マイクロプログラミング，産業図書(1977).

[8] 馬場敬信：マイクロプログラム記述言語とその処理系，情報処理，Vol.28,No.12,pp.1573-1584(1987).

[9] 当麻善弘：順序回路論，昭晃堂(1976).

[10] 渋井二三男,竹本宣弘：マイクロコンピュータインターフェース学習環境とその支援システム,電子情報通信学会誌，Vol.75,No.5,pp.488-495(1992).

[11] 神原弘之，安浦寛人：計算機教育用マイクロコンピュータの開発とその応用，情報処理，Vol.33,No.2,pp.118-127(1992).

[12] 末吉敏則：教育へのＦＰＧＡ応用例，情報処理,Vol.35,No.6,pp.519-529(1994).

[13] 柴山潔，新實治男：大学における計算機アーキテクチャの教育方法に関する考察，情報処理学会計算機アーキテクチャ研究会,100-4(1993).

[14] XILINX Inc.:Programmable Gate Array Data Book(1993).

[15] Aptix Coop.:Programmable Interconnect Data Book(1993).

第５章　あとがき

本論文では論理ゲートを構成要素とした非同期式順序回路の設計法からＦＰＧＡを用いたマイクロプロセッサの設計開発システムまでを論じた．この過程で次の点が明らかにされた．

非同期回路の設計では，ＳＲラッチの動作を解析し，ラッチの入力Ｓ，ＲがＳＲ＝０を満たしていれば，入力の静的ハザードは出力に影響しないことを明らかにした．ＳＲラッチによる非同期式順序回路の設計法で，ＳＲラッチの入力表からＳ，Ｒを求めるとつねにＳＲ＝０は満たされるので，Ｓ，Ｒをハザードフリーにする必要はない．したがってクリティカルレースがないように状態割当されていれば，非同期回路をＳＲフリップフロップを用いた同期回路と同じ方法で設計できることを示した．（第２章）

フリップフロップの設計では，まず，マスタスレーブ型Ｄフリップフロップとエッジトリガ型Ｄフリップフロップは同一の遷移表から導かれるので，論理的に同じものであることを示した．次に，広く使用されているＪＫフリップフロップの中に，典型的なマスタスレーブ型とエッジトリガ型の他に，クロックが１の間に入力が変化すると誤動作をする第３の型のフリップフロップがあることを示し，広く使用されている各種の回路がどの型に属するかを明確にした．

最後に，ＳＲラッチによる設計法を用いて，２種類の特殊なＪＫフリップフロップを設計し，既知の回路より優れた回路が得られている．（第３章）

マイクロプロセッサの設計では，プロセッサチップにＦＰＧＡを用いたマイクロプログラム制御方式のプロセッサを開発するシステムが構築されている．マイクロプログラム制御も結線制御も順序回路の１つの実現法であるという立場から，開発されたプロセッサの制御方式をデータプロセッサ部を変更せずに，マイクロプログラム制御方式から結線制御方式に変換する方法を検討し，いくつかの変換方法を示した．しかし，ここで示した方法でどの程度の回路が得られるかはわかっていない．これらの方法による変換の有効性を確かめることは今後の課題である．また，開発されたシステムは情報工学の計算機設計とアーキテクチャの実験実習教育に応用され，多数のプロセッサが開発され，教育用システムとしての有効性が確認されている．（第４章）

謝　　辞

京都大学在学中よりご指導ご鞭撻を頂いている萩原宏京都大学名誉教授に心より感謝の意を表します．

本論文は京都産業大学において行ってきた研究をまとめたものである．この間，御指導と有益な御援助を頂き，特に第２章，第３章の研究に協力して頂いた奥川峻史同大学工学部教授に感謝の意を表します．また，第４章の教育システムの構築に御協力頂いた同大学卒業生の岡田光博氏と古田勇次氏に感謝致します．

本研究をまとめるにあたり，懇切なるご指導を頂いた長谷川利治京都大学大学院工学研究科教授および矢島脩三同教授と茨木俊秀同教授に深く感謝致します．

本論文をまとめる契機を与えて頂いた宮野高明京都産業大学経営学部教授と，深いご理解とご支援を頂いた黒住祥祐同大学工学部長に心より感謝致します．